UNIVERSITÉ DU QUÉBEC

MÉMOIRE

PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIERES

COMME EXIGENCE PARTIELLE
DE LA MAITRISE EN ELECTRONIQUE INDUSTRIELLE

PAR
HASSAN EL-MOUSSAOUI

"STRATEGIE DE COMMUNICATION ENTRE UN SYSTEME DE CONTROLE ET UN RESEAU DE SENSEURS INTELLIGENTS DE STYRENE EN MILIEU INDUSTRIEL"

NOVEMBRE 1991

SOMMAIRE

Afin d'éviter les méthodes lentes pour la détection de la vapeur de styrène, qui sont basées sur l'analyse par chromatographie ou par spectrométrie, on a réalisé un système de communication bidirectionnelle, entre un réseau de capteurs intelligents qui est couplé à un micro-ordinateur. Ce système permet le contrôle et la détection rapide et in situ des vapeurs de styrène et des variables environnementales (température, pression) en milieu industriel. Ce système de communication est géré par un logiciel qui a été conçu spécialement dans le but de permettre à l'usager d'avoir accès à toutes les données provenant des capteurs connectés au réseau.

Les principaux objectifs poursuivis lors de la réalisation de ce système étaient la souplesse en termes de la programmation et de l'évolution topologique. Afin d'atteindre ces objectifs on a procédé par deux étapes:

La première concerne l'analyse d'une stratégie de communication bidirectionnelle permettant les échanges entre un micro-ordinateur central de commande et de décision, et un réseau de capteurs intelligents . Cette stratégie

aura pour fonction principale l'acquisition des données provenant des différents points d'accès reliés au réseau.

La deuxième partie traitera de la structure du logiciel du micro-ordinateur central. Cette structure devra permettre à l'usager d'avoir accès à chacun des points reliés sur le lien et devra permettre l'acquisition des données. De plus, la procédure d'acquisition devra pouvoir aisément être modifiée en fonction des besoins de l'usager et ce, même après l'installation du système.

Un banc d'essai a été construit afin de vérifier la performance de ce système de communication dans un environnement réel. Deux approches distinctes ont alors été évaluées: La première mettait l'emphase sur l'accessibilité à des composants matériels spécialisées alors que la deuxième préconisait une approche logiciel afin de minimiser les coûts du matériel.

La structure globale de ce système de communication, à multiples points d'accès, possède les caractéristiques suivantes:

- la possibilité d'acquisition de données, de transfert d'information et de contrôle de divers points d'échange de façon automatique et reprogrammable;

- la flexibilité de cette procédure d'acquisition qui va permettre une simple modification des modules de communication;

- la simplicité d'installation des différents points sur le réseau;

- l'adoption d'une topologie simple de réseau afin de permettre à un utilisateur non-spécialiste de localiser aisément toute source de défaillance.

REMERCIEMENTS

Qu'il me soit permis d'exprimer mes sincères remerciements à tous ceux qui m'ont apporté leur soutien indéfectible tout au long de ce travail.

Messieurs Phieu Le-Huy et Demagna Koffi ont dirigé ce mémoire, le premier en tant que directeur et le second comme co-directeur. Malgré leurs multiples occupations, ils ont toujours su m'apporter une assistance technique soutenue et un support chaleureux. Je les remercie très sincèrement.

Je n'oublie pas M. James Agbébavi et M. Daoud Ait Kadi qui ont toujours su me soutenir lors de mes moments de découragement. Leur détermination et leur enthousiasme ont été un facteur de motivation pour l'accomplissement de ce travail.

Je suis très reconnaissant envers M. Serge Deblois, pour ses conseils judicieux, ses suggestions et son assistance technique au laboratoire. Très amicalement merci.

Signalons que cette étude a été menée grâce à la subvention RS-89-09 de l'Institut de Recherche en Santé et Sécurité au Travail du Québec (IRSST)

Les derniers, mais non les moindres. Je rends hommage à ma chère famille pour son soutien moral. Ils se sont adaptés avec un courage et une patience exemplaires aux circonstances parfois difficiles, imposées par cette recherche. Leur amour pour moi est le fruit de ce travail. Je leur dis fraternellement merci.

LISTE DES FIGURES

Fig 1.1 Structure centralisée de capteurs intelligents 9

Fig 2.1 Liaison de données 14

Fig 2.2 Liaison parallèle 17

Fig 2.3 Liaison série 18

Fig 2.4 Liaison en étoile 22

Fig 2.5 Liaison multipoint en anneau 22

Fig 2.6 Liaison multipoint en bus 23

Fig 2.8 Liaison de données en mode de commande. 31

Fig 2.9 Source et puits d'informations. 31

Fig 2.10 Contrôle d'une liaison point à point par sélection 32

Fig 2.11 Contrôle d'une liaison point-à-point par invitation à émettre 33

Fig 2.12 Liaison bidirectionnelle symétrique 34

Fig 2.13 Liaison bidirectionnelle dissymétrique 35

Fig 2.14 Mode synchrone 37

Fig 2.15 Mode asynchrone. 38

Fig 2.16 Format du message émis par les capteurs intelligents 40

Fig 3.1 Unité de communication asynchrone. 43

Fig 3.2 Formatage des données 46

Fig 3.3 Ligne bien adaptée 54

Fig 3.4 Ligne non adaptée (Z#Z0 réflexion) 54

Fig 3.5 La fréquence de modulation 56

Fig 3.6 Emetteur de ligne 57

Fig 3.7 Récepteur de ligne 57

Fig 3.8 Ligne équilibrée 59

Fig 3.9 Ligne déséquilibrée 59

Fig 3.10 Schema bloc du lien de communication 61

Fig 4.1 Module d'initialisation pour la station de contrôle 70

Fig 4.2 Module d'initialisation pour le capteur intelligent 71

Fig 4.3 Structure minimale du module d'acquisition de données 73

Fig 4.4 Structure du programme au niveau du capteur intelligent 79

Fig 4.5 Structure du logiciel pour plusieurs capteurs 81

Fig 4.6 Schéma bloc du montage expérimental 82

Fig 4.7 Montage du bain thermostaté 83

LISTE DES TABLEAUX

Tableau 2.1 Caractéristiques de différents supports physiques 20

Tableau 2.2 Mode d'exploitation d'une liaison 25

Tableau 3.1 Caractéristiques électriques des convertisseurs de niveaux [36] 53

TABLE DES MATIERES

SOMMAIRE I

REMERCIMENTS IV

LISTE DES FIGURES V

LISTE DES TABLEAUX VII

TABLE DES MATIERES VIII

CHAPITRE I 1

INTRODUCTION 1

I-1 Généralités 1

I-2 Problématique 5

I-3 Objectif de l'étude 7

I-4 Disposition du mémoire 11

CHAPITRE II 13

ANALYSE D'UN LIEN DE COMMUNICATION POUR LES CAPTEURS INTELLIGENTS 13

II-1- Analyse d'une liaison de données 13

II-1-1- Aspect physique d'une liaison de données 14

II-1-1-1- Branchements 15

II-1-1-1-1- Interface parallèle 16

II-1-1-1-2- Interface série 18

II-1-1-2- Support physique d'interconnexion 19

II-1-1-3- Structure de la liaison 21

II-1-1-4- Mode d'exploitation de la liaison 23

II-2- Aspect logique de la liaison de données 24

II-2-1- Fonctionnement et gestion de la liaison multipoint 25

II-2-1-1- Statégie de communication par diffusion aléatoire 26

II-2-1-2- Stratégie de communication par invitation à émettre et à recevoir 34

II-3- Protocole pour le lien de communication 35

II-3-1- Analyse du format utilisé pour les messages 36

II-3-2- Format des messages utilisé 38

CHAPITRE III 41

MATERIEL UTILISE ET SON OPERATION 41

III-1- Carte de communication série 42

III-1-1- Formatage des données 43

III-1-2- Fonctions de l'émetteur 45

III-1-3- Fonctions du récepteur 47

III-2- Entrées/sorties du capteur intelligent 49

III-2-1- Description générale 49

III-2-2- Fonctionnement de l'SCI 50

III-3- Convertisseur de niveau 52

III-3-1- Normes électriques 52

III-3-2- Impédance caractéristique 53

III-3-3- Émetteur et récepteur de ligne 56

III-3-4- Émetteur et récepteur de ligne 56

III-3-5- Émetteurs et récepteurs différentiels 58

III-4- Circuit réalisé pour le lien de communication 60

III-5- Mise en oeuvre de la liaison de données en milieu industriel 62

CHAPITRE IV 64

CONCEPTION DU LOGICIEL DE COMMUNICATION 64

IV-1- Le module d'initialisation 65

IV-1-1- Initialisation de l'UART 65

IV-1-2- Initiatisation de l'SCI 69

IV-2- Le module de communication 72

IV-2-1- Structure du logiciel de la station de contrôle 72

IV-2-2- Structure du logiciel du capteur intelligent 77

IV-3- Evaluation du logiciel de communication 82

IV-3-1- Étude expérimentale 83

IV-3-1-1- Manipulation par le jeu des manomètres 85

IV-3-1-2- Manipulation par injection directe 86

IV-4- Résultats expérimentaux 88

CHAPITRE V 90

CONCLUSION 90

BIBLIOGRAPHIE 93

ANNEXE. 1 Listing du programme de simulation d'un bus auquel sont reliées quatre stations émettrices de messages 98

ANNEXE .2 Listing du programme de communication implanté au niveau du micro-ordinateur 114

ANNEXE. 3 Listing du programme implanté au niveau du capteur intelligent 145

ANNEXE. 4 Fiche technique du microcontrôleur M68HC11 151

ANNEXE. 5 Schéma fonctionnel de ACE-8250 154

ANNEXE. 6 Fiche technique du RS-232 et du RS-422 157

ANNEXE. 7 Résultats expérimentaux 162

CHAPITRE - I

INTRODUCTION

I-1 Généralités

Les moyens de communication ont évolués profondément au cours de l'histoire, notamment au cours des deux cents dernières années. Le premier moyen mis à notre disposition réside évidemment dans le langage oral ou gestuel [3]. Cependant, ces limites sont atteintes dès que les interlocuteurs ne sont pas physiquement au même endroit.

Il fallut donc imaginer d'autres méthodes de transmission. A partir du moment où les informations ont pu être traduites par écrit, le courrier est devenu le moyen le plus naturel pour effacer les distances.

L'histoire fourmille d'exemples plus ou moins ingénieux des systèmes assurant le transfert d'informations d'un point à un autre. Cela va du simple messager chargé de transporter une missive écrite jusqu'aux sémaphores, dont les pavillons codés étaient déchiffrables d'un bateau à l'autre, en passant par les signaux de fumée des Indiens ou les tam-tams de certaines tribus africaines. Ces diverses méthodes n'étaient pas parfaites, car elle n'apparaissaient pas toujours d'une fiabilité suffisante et ne permettaient

généralement pas de transmission à une longue distance. Il faudra finalement attendre le 19e siècle pour voir apparaître des techniques bouleversant réellement le paysage des communications et entraînant la disparition complète de la plupart des méthodes de communication utilisées précédemment [3].

Ces nouveautés se caractérisent essentiellement par l'utilisation de l'énergie électrique (ou électronique) comme support de transmission. La modulation d'un signal électrique est utilisée pour transporter des messages. L'énergie devient porteuse d'informations, au même titre que le messager ou que le moyen de transport traditionnel, mais avec une vitesse nettement supérieure. Or les moyens de l'ordinateur et de l'informatique utilisés pour le traitement sont plus rapides et plus efficaces.

D'autre part, les possibilités technologiques de la micro-électronique ont contribué à l'apparition du concept de capteur intelligent. En dotant le transmetteur associé aux transducteurs d'un organe de calcul interne (microprocesseur ou autre), les performances métrologiques et fonctionnelles d'un capteur sont considérablement accures.

Les caractéristiques nouvelles, d'un capteur intelligent, sont principalement un gain important sur la précision et la crédibilité des mesures, l'élaboration de signaux numériques et la possibilité de communication avec un réseau local ou un calculateur. Cette communication avec le réseau ou le système d'acquisition, permettant l'envoi d'information (mesures, états, contrôles internes...) et la réception d'ordres ou de données, le tout sous forme numérique.

Avant même l'apparition des premiers ordinateurs, la transmission de données informatiques fut expérimentée. En 1940 le mathématicien américain George Stibitz eut l'idée d'interroger à distance une calculatrice de sa conception. Pour cela, des techniciens des laboratoires Bell imaginèrent un terminal permettant des liaisons à partir d'un réseau de téléscripteurs. Pourtant ce procédé demeurait particulièrement rustique et lent. Cependant, à cette époque, la vitesse de transmission n'était pas un élément déterminant puisque les calculateurs étaient eux même particulièrement lents [3].

À la fin des années 40 et avec l'arrivée des ordinateurs, tous ces méthodes de communication allaient changer. Ces ordinateurs sont des matériels équipés d'une unité centrale, chargés des traitements de données et des commandes d'opérations. Leurs mémoires permettent le stockage temporaire ou permanent d'informations, et enfin leurs périphériques autorisent le dialogue entre l'unité centrale et les utilisateurs humains . La volonté de relier les systèmes à microprocesseurs entre eux n'est pas récente, puisqu'elle remonte aux années cinquante, avec la mise en place du système de défence aérienne SAGE (Semi-Automatic Ground Environment), qui reliait de nombreuses stations radar, situées aux Etats-unis et au Canada, à vingt-septs centre régionaux [3].

Alors, faire communiquer des capteurs intelligents (systèmes à microprocesseur) est un souci croissant qui entraîne une inversion de tendance. Aujourd'hui, un capteur n'est plus rien s'il n'est pas relié à un réseau ou à une centrale d'acquisition. Les réseaux les plus performants ne doivent pas se contenter de transférer des suites d'éléments binaires, il faut

encore que cette succession de bits ait la même signification pour l'émetteur et pour le récepteur. Le problème est identique à celui de deux personnes s'exprimant dans deux langues distinctes. Pour se comprendre il faut soit que l'une connaisse la langue de l'autre, soit que toutes deux soient en mesure de parler une troisième langue commune. Au niveau des réseaux informatiques ces problèmes de compatibilité furent réglés par l'introduction de divers protocoles qui ne sont rien d'autre que des règles strictes régissant les échanges. C'est ainsi qu'ils détailleront la forme des informations amenées à transiter sur le réseau.

Le premier avantage de mariage, entre informatique et communication, réside dans la suppression des supports physiques de l'information. C'est ainsi que les transferts à l'aide de supports magnétiques (disques ou disquettes) laissent progressivement place aux transmissions de données.

Le second intérêt est lié à un accroissement important de la rapidité de communication. Les transferts se réalisent en temps réel. Le destinataire prend connaissance de l'information au moment même où elle a été expédiée.

Le dernier avantage de ce mariage est la suppression des redondances d'information. C'est un élément aussi fondamental que les deux précédents. En effet, relier ces capteurs intelligents par un réseau permet de mettre à la disposition des intervenants une seule entité. Cette unicité permet d'éviter les méthodes lentes (la détection de la vapeur de styrène, par exemple) qui sont basées sur l'analyse par chromatographie ou par spectrométrie.

I-2 Problématique

L'incendie, en septembre 1988, des barils de polychorobiphenyle (PBC) à Saint-Basile-le-Grand, Canada, n'a fait qu'accentuer l'épineux problème des détections rapides des toxines et polluants toxiques auxquels sont exposées des populations. Les études actuelles couvrant les polluants gazeux dans l'air et dans les précipitations [1], le méthane [2], les aérosols [3], ou les oxydes de soufre [4] sont monnaie courante mais s'occupent surtout des transports atmosphériques de ces déchets sur de longues distances. L'un ou l'autre de ces polluants toxiques a l'histoire du styrène ou de ses dérivées.

Le dégagement des vapeurs de styrène dans l'air peut se produire au cours des opérations de synthèse de styrène et de ses polymères [1], pendant la fabrication de fibres de verre [6], lors de bris de machines ou d'epandages accidentels [7] ou durant le travail des matériaux à base de polymères de styrène [7]. L'absorption de styrène, par les travailleurs des industries concernées, se fait par respiration sous forme de vapeurs [5] et par contact cutané sous forme liquide [10]. L'émanation de styrène contamine donc l'air ambiant des usines et contribue aussi à la pollution atmosphérique à l'extérieur des industries en question.

Le métabolisme du styrène dans l'organisme humain se traduit en la présence d'acides phenylglyoxylique et mandélique dans l'urine [11] et de

styrène glycol dans le sang [12]. Cependant la grande partie du styrène infiltré dans le système biologique s'accumule dans le foie, les reins et dans les tissus adipeux [13]. La toxicité des vapeurs de styrène et de ses dérivés a été étudiée [14], la limite de tolérance de l'organisme est de 100ppm [16].

Les méthodes connues de détection du styrène sont basées sur l'analyse par chromatographie en phase gazeuse [15], [16] en phase liquide [17] ou par spectrométrie [13]. Ces méthodes d'analyse sont très fiables mais elles utilisent des échantillons prélevés sur les lieux ou des échantillons de produits biotransformés dans le corps de travailleurs [9]. Elles donnent des résultats après de longs délais et l'échantillonnage est différent selon les pièges utilisés [18]. L'analyse par spectrométrie au laser sur les lieux promet [19], mais la technologie reste à développer. L'ensemble des systèmes de détection employé est grand mais il constitue des solutions à posteriori. Alors, il devient ainsi indispensable de pouvoir détecter rapidement, in situ, les vapeurs de styrène avant une exposition prolongée des travailleurs. La venue des capteurs intelligents pourrait être exploitée pour ce but.

Le développement et la recherche dans le domaine de la micro-électronique [23], ainsi que les exigences demandées en ce qui concerne la qualité des mesures et des informations dans les systèmes de production industriels [24], ont donnée naissance à une nouvelle génération de capteurs nommés "capteurs intelligents" [25]. Ils peuvent être exploités pour remédier au problème de styrène.

Il s'avère important de pouvoir détecter rapidement et in situ les vapeurs de styrène au moyen d'un réseau de capteurs intelligents couplé à un

système intégré de contrôle. Ces capteurs intelligents, qui possèdent un pouvoir de contrôle décisionnel sur les mesures de styrène ainsi que les paramètres environnementaux (température, pression et humidité) de l'atmosphère avoisinant, sont physiquement distribués selon une topologie à déterminer en fonction des caractéristiques du milieu à contrôler. De plus, ces capteurs sont numériquement adressables et peuvent communiquer avec l'unité centrale de supervision selon une priorité de décision pré-établie. A ce stade, il est primordial d'élaborer une hiérarchie de décision et d'adopter une stratégie de communication efficace entre divers éléments du système intégré de contrôle.

I-3 Objectif de l'étude

La présente étude vise l'étude et le développement d'une structure d'échange d'information d'un réseau à multiples points d'accès. Cette étude se subdivise en deux parties principales:

La première concerne l'analyse d'une stratégie de communication bidirectionnelle permettant les échanges entre un micro-ordinateur central de contrôle, de décision, et un réseau de capteurs intelligents (Fig 1.1). Cette stratégie aura comme fonction principale l'acquisition des données provenant des différents point d'accès reliés au réseau.

La deuxième partie traitera de la structure du logiciel du micro-ordinateur central. Cette structure devra permettre à l'usager d'avoir accès à chacun des points reliés sur le lien et devra permettre l'acquisition des données. De plus, la procédure d'acquisition devra pouvoir aisément être modifiée en fonction des besoins de l'usager et ce, même après l'installation du système.

On a constaté qu'il n'existe que très peu de littérature technique concernant la réalisation de ces deux parties principales. Par contre, la majorité des développements ayant rapport à une telle étude est plutôt orientée vers l'acquisition de données ou vers le développement de réseaux locaux.

Ainsi, lors de l'analyse des systèmes d'acquisition conventionnels, il est apparu que la majorité d'entre eux ne sont performants que lorsque restreints à une application spécifique [27,28], par ailleurs, ceux qui possèdent une plus grande flexibilité nécessitent généralement l'achat d'un matériel spécialisé très dispendieux [29, 30, 31].

Dans le cas des réseaux locaux, ce n'est pas la flexibilité qui est en cause mais plutôt l'inefficacité des protocoles utilisés lorsque les messages échangés sont courts (blocs de 16 bits ou moins) [32,33]. En effet, ces systèmes n'ont pas été développés pour une telle application. Par ailleurs, le coût impliqué par l'utilisation de ces systèmes est inacceptable (par exemple le système PAD coute 12,000 $ minimum). Certains manufacturiers ont tenté de les utiliser et de les adapter à de petits réseaux [34].

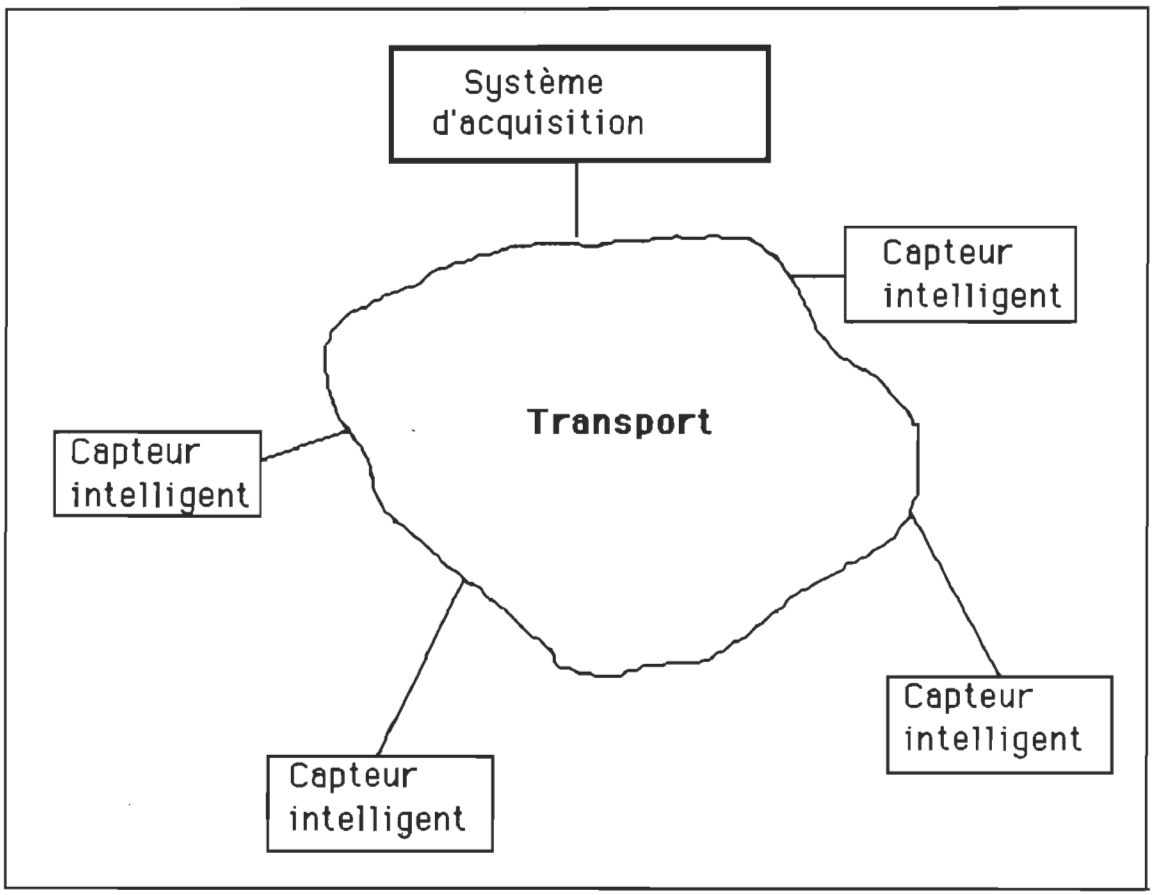


Fig. 1.1. Structure centralisée de capteurs intelligents

En fait, seul un choix judicieux de certaines caractériques puisées dans chacun de ces deux domaines, systèmes d'acquisitions de données et réseaux locaux, peut permettre de satisfaire adéquatement aux objectifs suivants:

1 - le développement d'une stratégie de communication bidirectionnelle, pour permettre l'échange entre un micro-ordinateur central de contrôle et un réseau de capteurs intelligents,

2 - la réalisation d'une procédure d'acquisition qui va être entièrement accomplie par le micro-ordinateur, ce dernier n'ayant qu'à recueillir les différents résultats et à faire ensuite le traitement désiré,

3 - cette procédure d'acquisition doit être flexible et simple, pour permettre des modifications dans le futur si nécessaire,

4 - l'installation des différents capteurs intelligents sur le réseau devra être simple et facile,

5 - l'adoption d'une topologie simple de réseau permettra à un utilisateur non spécialisé de localiser aisément toute source de défaillance,

6 - l'élaboration d'un protocole fiable permettra de minimiser les risques d'erreurs d'interprétation lors de la transmission,

7 - la conception d'un logiciel de communication qui va permettre un accès facile à toute information du réseau.

I-4 Disposition du mémoire

L'ordre de présentation des différents chapitres de ce mémoire doit refléter assez fidèlement le cheminement suivi par l'auteur afin de permettre aux lecteurs d'évaluer chacun des choix (ou compromis) réalisé à chaque étape importante du travail.

Le **premier chapitre** est consacré à une brève introduction sur la communication, les capteurs intelligents et surtout sur les problèmes causés par les vapeurs de styrène.

Suit le **chapitre II** qui analyse différentes stratégies de communication pour une structure d'échange du réseau à multiples points d'accès. Cette analyse est appuyée par une simulation bien détaillée. Ensuite on fait une analyse et un choix judicieux des différents éléments de notre système de communication comme:

- le choix de branchements;

- le choix du support physique d'interconnexion;

- le choix d'une topologie;

Enfin le protocole choisi pour la communication est présenté.

Le **chapitre III** expose l'aspect matériel du système de communication, comme les convertisseurs de niveaux, la carte de communication série et l'adaptation de la ligne de transmission.

Le **chapitre IV** consistera en une analyse des différents modules constituant le logiciel de communication. Ce logiciel va gérer le réseau de capteurs intelligents, avec une flexibilité et une facilité d'accès aux différents points du réseau. Puis on présente les résultats obtenus à partir de l'étude expérimentale. Cette étude expérimentale servira aussi à l'évaluation du logiciel conçu pour cette fin. Le mémoire finit sur une récapitulation des

principaux résultats, tire les conclusions qui s'en dégagent et propose les suggestions pour la suite du projet.

CHAPITRE II

ANALYSE D'UN LIEN DE COMMUNICATION POUR LES CAPTEURS INTELLIGENTS

II-1- Analyse d'une liaison de données

Pour être acheminée, toute information doit être mise sous forme de symboles. La signification précise de ces symboles est évidemment fondamentale, mais elle ne découle que d'une convention entre l'émetteur du message et le distinataire (récepteur). Toutefois, dans la pratique, chaque organe impliqué dans une transmission de données est suscuptible d'être mis en relation avec plusieurs autres organes; d'autre part, la traduction de l'information devra nécessiter une operation "physique" (même si elle est réalisée par un logiciel).

Dans le cas présent, l'information est émise ou reçue par un équipement terminal de traitement de données (ETTD) souvent appelé terminal, mais qui peut être un ordinateur ou comportant des fonctions de traitement (Fig 2.1).

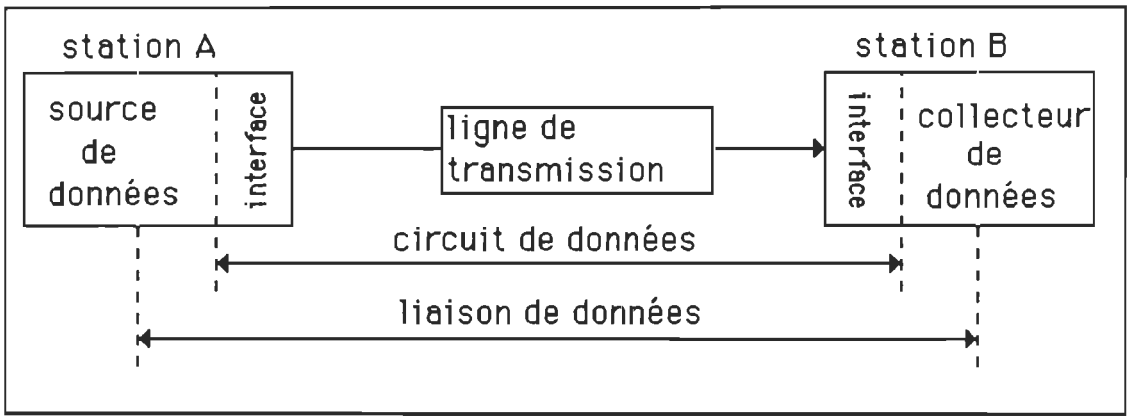


Fig 2.1 . **Liaison de données**

Selon la définition de "Data Link", livre rouge du CCITT [35], la liaison de données est l'**ensemble des installations terminales et du réseau d'interconnexion associé qui fonctionne dans un mode particulier permettant l'échange d'informations entre les installations terminales.**

La liaison de données, ainsi définie, peut être considérée sous deux aspects:

- Un physique, lié au circuit de données et à la transmission de données;

- Un logique, lié à la commande de la liaison et à la coordination du transfert des données pour le rendre sûr et efficace.

II-1-1- Aspect physique d'une liaison de données

Le circuit de données est l'ensemble des moyens qui permettent l'échange de données entre les deux stations. Un tel circuit comporte en général les branchements et le support de transmission [35].

La communication informatique suppose une liaison permanente ou temporaire entre les différents matériels utilisés. Cette connexion informatique requiert naturellement un certain nombre de branchements. La nature différente des informations circulant dans un ordinateur (ou systèmes à micoprocesseurs) et sur des lignes de comunication conduit à utiliser certains accessoires matériels spécifiques aux communications numériques.

II-1-1-1- Branchements

Pour réaliser ces branchements, il faut avoir des interfaces entre micro-ordinateur et périphériques qui sont destinées à relier les bus des microprocesseurs aux équipements périphériques.

Cette interface assurera la transmission d'une information de ou vers l'environnement, aura aussi en général une certaine "intelligence" qui lui permettera de coder, décoder, mettre en forme, traduire et adapter l'information, et devra assurer la communication, c'est la transmission de l'information.

La plupart des micro-ordinateurs disposent de deux catégories de liens avec le monde extérieur: la liaison parallèle ou la liaison série; donc il faut examiner et analyser les interfaces parallèles et les interfaces séries pour faire un choix approprié de branchement entre les capteurs à connecter et le micro-ordinateur.

II-1-1-1-1- Interface parallèle

Dans une transmission parallèle, il faut autant de fils ou de connexions qu'il y a de bits par mots. Les bits sont transmis en parallèle, mais les mots en série, l'un après l'autre. La figure 2.2 symbolise une liaison parallèle sur 8 bits où les huits connexions sont représentées. Cet ensemble de fils conducteurs électriques montés en parallèle assure uniquement le cheminement de l'information entre les deux systèmes: c'est une voie de communication. Il est nécessaire de mettre au point une interface pour chaque système afin d'adapter l'information transmise ou reçue. Cette interface est composée d'une partie matérielle et d'une partie logicielle.

Les interfaces de type parallèle sont les moins complexes, certaines interfaces parallèles peuvent être constituées d'un seul circuit intégré TTL. Par ailleurs, certaines techniques d'interfaces parallèles exigent des circuits beaucoup plus élaborés.

Les interfaces parallèles peuvent être classées selon deux critères:

- le nombre de bits transmis en parallèle par l'interface (largeur du bus de don-nées), qui peut varier de 8 à 32 bits;

- le type d'asservissement employé pour faire circuler l'information sur les lignes de données.

La complexité des interfaces parallèles croît avec l'augmentation du nombre de lignes d'asservissement. En outre, les interfaces simples sont conçues pour ne relier qu'un seul périphérique. Pour relier un second

dispositif à notre micro-ordinateur, il faut qu'on double les circuits d'interfaces. Ceci est nécessaire parce que les interfaces simples ne possèdent pas des mécanismes de sélection de dispositifs.

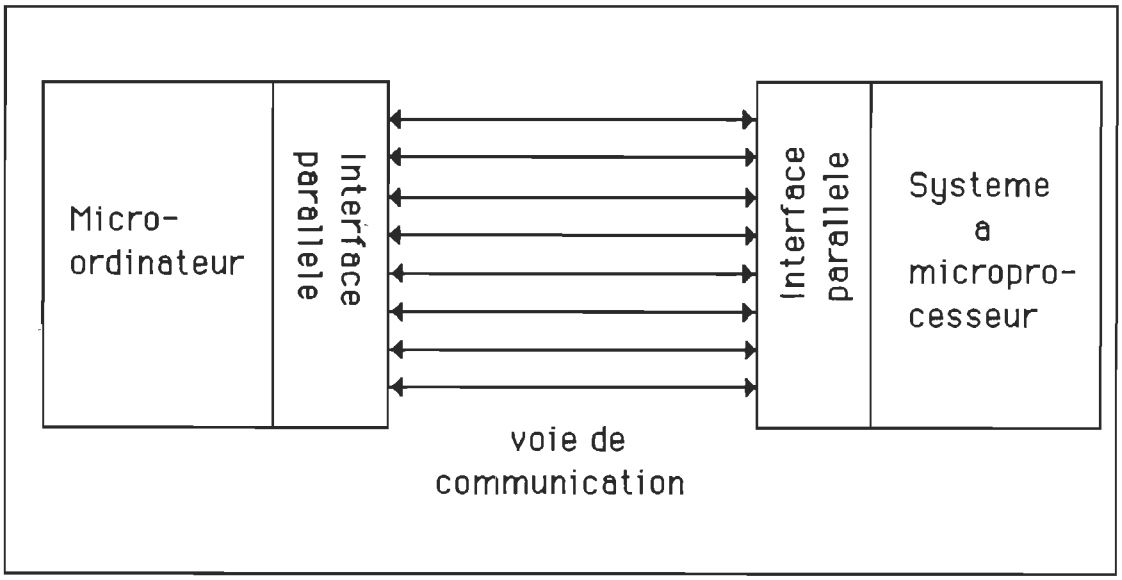


Fig 2.2. **Liaison parallèle**

En premier lieu on a pensé à une interface numérique normalisée pour raccorder des capteurs, c'est-à-dire l'interface IEEE-488 [36]. Elle n'est pas universelle mais suffisamment riche pour répondre à un grand nombre d'applications. Après l'étude des besoins, les limites proposées par les organismes de normalisation ont été fixées sur des critères tels que le nombre de stations, la vitesse de transmission, la distance maximale de transmission des messages, la souplesse, la compatibilité et le coût. Les caractéristiques sont donc les suivantes:

- transmission du type bits parallèles/octets série;

- au maximum 15 stations connectées sur le même bus;

- la distance maximale de transmission des messages sur le bus est de 20 mètres;

- la vitesse maximale de transmission est de10 Moctets/s;

- une procédure efficace d'échange entre un émetteur et un récepteur;

- une grande facilité de raccordement.

D'après ces caractéristiques, on remarque qu'on est limité par le nombre de stations et par la distance de la liaison. Par contre, si on choisit les interfaces parallèles programmables (exemple: Intel 8255), le problème qui se pose c'est qu'on doit ajouter un grand nombre des circuits spécifiques qui permettent de transformer les niveaux TTL en niveau de ligne et réciproquement. En plus, le transport de chaque bit sur chaque fil, rend le coût d'une liaison de plusieurs fils en parallèle plus prohibitif. L'interface série fournit une solution à un coût plus faible.

II-1-1-1-2- Interface série.

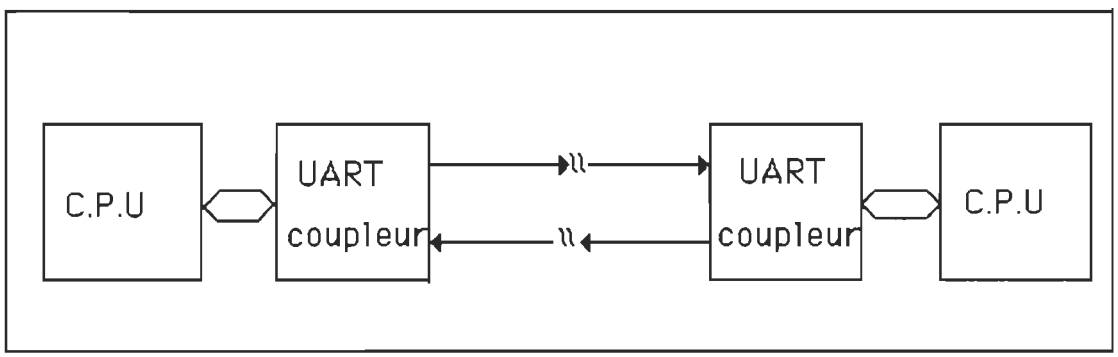


Fig 2.3. **Liaison série**

Le principe de base de la transmission de données série est le transfert des données sur un seul fil (Fig 2.3). L'interface permettant de réaliser la transformation de signaux reçus de l'unité centrale CPU est appelée coupleur; il s'agit le plus souvent d'un circuit appelé l'UART (Universal Asynchronous Receiver/Transmitter). Les informations fournies sortent ou entrent dans l'UART étant en logique TTL. Alors, il est nécessaire d'utiliser des circuits spécifiques permettant de transformer ces niveaux, en niveaux de ligne ±12V et réciproquement.

Cette interface sera choisie comme un élément du support de transmission dans cette étude. Puisque l'utilisation d'un seul fil pour transférer les données demeure l'atout principal qui justifie ce choix, donc elle sera plus détaillée dans le chapitre qui suit.

II-1-1-2- Support physique d'interconnexion

Lors de la conception d'un système de communication, le choix du support physique est notamment influencé par les performances que l'on attend du système à réaliser. Les principaux types de supports utilisés sont les fils métalliques, le câble coaxial et la fibre optique. Il ont des caractéristiques très différentes [36] (voir Tableau 2.1), tant en matière de débit escompté que d'encombrement, d'atténuation ou de coût.

La fibre optique est un support d'avenir très performant; on l'utilise dans des environnements perturbés tels que les entreprises industrielles car elle n'est pas sensible aux parasites éléctriques. Mais, la fibre optique a aussi

quelques inconvénients. L'inconvénient principal reste encore aujourd'hui le raccordement qui n'atteint pas la souplesse de la paire torsadée ou du câble coaxial. Les dérivations et les liaisons multipoints sont relativements difficiles à mettre en oeuvre. En terme de coût la fibre optique est le support le plus onéreux.

Tableau 2.1. **Caractéristiques de différents supports physiques**

Support de transmission	Portée	Débit	Facilité topologique	Immunité au bruit	Coût
Paire torsadée	10 km	1 Mb/s	grande	faible	faible
Câble coaxial	3 km	25 Mb/s	grande	modérée	raisonnable
Fibre optique	500 km	+100Mb/s	modérée	exellente	élevé

En plus des avantages de débit, de portée, le câble coaxial permet la réalisation simplifiée de la liaison multipoint; il suffit dans ce cas de faire une connexion passive sur le câble refermé sur son impédance caractéristique (50 ou 75 ohms), on peut le couper sans perturber les signaux. De plus, il est peu sensible aux parasites. Son coût reste l'inconvénient majeur.

Enfin, notre choix s'arrête sur le support physique le plus simple: la paire torsadée. Il est réalisé à partir de paires de fils électriques (0.2 mm à 1 mm

de diamètre). Elle permet des transmissions de données numériques à une vitesse d'environ 1Mbits/s sur une distance de 1km. Les pricipaux avantages de ce type de support sont: la simplicité de connexion et le faible coût. Pour contourner l'inconvénient de la limitation du débit, point moins important pour notre application portant sur une courte distance (environ 1km), qui est dû à une forte atténuation du signal et la relative sensibilité aux parasites, on utilise des paires torsadées blindées. Les caractéristiques électriques d'une paire de fils, comme l'impédance caractéristique, le rapport vitesse-longueur de la ligne et l'immonité au bruit, seront étudiés à fond à la section 4.3.

II-1-1-3- Structure de la liaison.

Des variétés de structures possédant des atributs appropriés aux réseaux de communication ont été identifiées par des spécialistes en conception des résaux; les trois retenues sont: les structures en étoile, en anneau et en bus. L'avantage de la structure en étoile (Fig 2.4) est d'éliminer pour chaque noeud sur le réseau le besoin de prendre des décisions routinières en les localisant au noeud central, il est clair cependant, qu'un tel réseau dépend du bon fonctionnement du noeud central de même que de sa capacité de supporter les conversations simultanées des stations. Son inconvénient majeur est le manque de fiabilité de cette topologie; en cas de panne du centre, l'ensemble du réseau est en panne. La structure en étoile est une configuration appelée aussi liaison point-à-point; par contre les structures anneau et bus sont des liaisons multipoints car on peut connecter plusieurs stations à un équipement central.

Dans la configuration anneau (Fig 2.5), le support relie toutes les stations de manière à former un circuit fermé. L'information circule dans une seule direction le long du support de transmission. Le problème de fiabilité de ce type de configuration lui a valu de nombreuses critiques [38]. En effet, la rupture de l'anneau ou la défection d'un noeud actif paralyse le trafic du réseau.

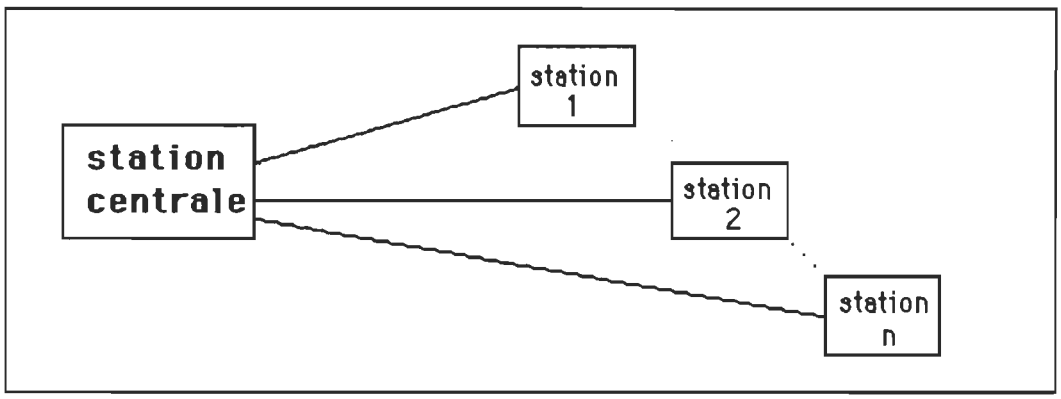


Fig 2.4. **Liaison en étoile**

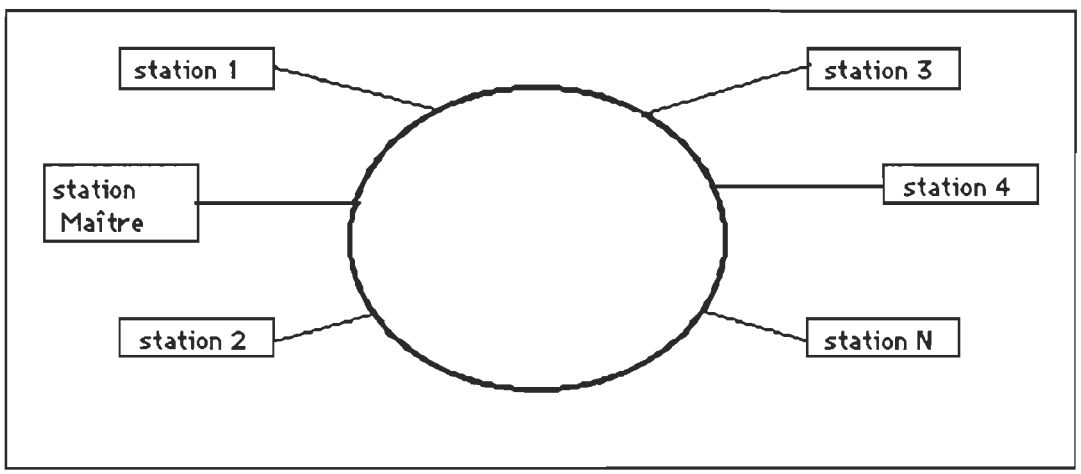


Fig 2.5. **Liaison multipoint en anneau**

La configuration à retenir pour cette étude est le bus (Fig 2.6) . Dans cette structure, l'ensemble des stations est raccordé sur une liaison physique commune. Si les données émises par la station centrale sont reçues simultanément par toutes les stations connectés sur le bus, les transmissions vers la station centrale ne peuvent s'éffectuer que par séléction ou par scrutation, d'une manière ordonnée [38].

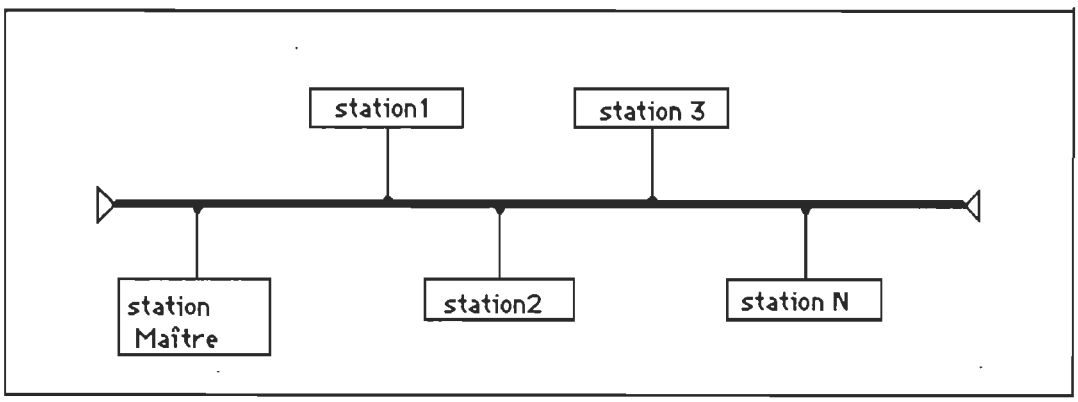


Fig 2.6. **Liaison multipoint en bus**

II-1-1-4- Mode d'exploitation de la liaison.

Les possibilités offertes par le support physique de transmission sont: circuit simplex, semi-duplex ou duplex. Les liaisons de données organisent les échanges sur le support physique selon un mode d'exploitation logique qui définit le sens de circulation des informations. Le mode d'exploitation logique ne sera pas nécessairement identique au mode de fonctionnement physique et peut être soit unidirectionnel, soit bidirectionnel à l'alternat, soit bidirectionnel simultané, [35]. Il faut que la liaison physique et la liaison

logique soit compatibles. Il pourrait sembler illogique d'exploiter une liaison avec des possibilités réduites par rapport à celle du canal physique, par exemple en travaillant à l'alternat sur un canal duplex. En fait, ces modes d'exploitation sont très souvent employés, car ils simplifient la procédure des échanges entre les stations. Le Tableau 2.2 résume les possibilités offertes par des liaisons physiques, qui va nous aider à bien choisir la possibilité d'une liaison logique.

II-2- Aspect logique de la liaison de données.

Jusqu'à present, on a fait la description du matériel. Cependant, si la liaison définie permet de transférer des éléments binaires d'un point de la liaison à un autre, il est évident qu'il faut une logique pour effectuer correctement ce transfert, pour l'interpréter et pour faire face aux situations anormales. Cette logique constitue la procédure de commande (ou protocole de transmission) pour la liaison de données.

La procédure de commande joue le rôle d'intermédiaire, d'interface logique entre la transmission et le traitement de l'information. A ce titre, la procédure de commande doit inclure, en plus de la gestion proprement dite de la liaison de données, une gestion d'interface avec la transmission et une gestion d'interface avec le traitement. Ce rôle d'interface impose aux procédures de commande diverses contraintes que nous analyserons dans la suite.

Le but principal de la procédure de commande est de transférer de l'information d'une extrémité de la liaison à une autre. Pour cela, il faut

structurer les données, les délimiter (c'est-à-dire indiquer le début et la fin), les identifier.

II-2-1- Fonctionnement et gestion de la liaison multipoint.

La liaison de données gère les échanges d'information entre des stations connectées à un même canal de transmission. Cet échange d'information entre stations et une station centrale peut être, soit par contrôle du trasfert d'information par invitation à émettre et invitation à recevoir, ou par contrôle par diffision aléatoire.

Tableau 2.2 . **Mode d'exploitation d'une liaison**

liaison logique / liaison physique(circuit)	Mode d'exploitation unidirectionnelle	mode d'exploitation bidirectionnelle	
		à l'alternat	simultané
point à point	X	X	X
multipoint	X	X	limité à 2 stations
simplexe	X		} *
semi-duplex	X	X	
duplex	X	X	X

* la voie secondaire, encore appelée voie de supervision, peut permettre, dans le cas de circuits non duplex, de transmettre simultanément l'information dans un sens et un signal, par exemple un accusé de réception, dans l'autre sens, mais à vitesse plus faible. Cette méthode n'est pas très fréquemment utilisée.

II-2-1-1- Statégie de communication par diffusion aléatoire

La méthode de contrôle par diffision aléatoire consiste à ce que toutes les stations trasmettent des données vers la station centrale, sur un même canal, de façon aléatoire. Ainsi, aucune station ne contrôle le transfert d'information. L'instant de début de transmission peut être librement choisi par chacune des stations émettrices ou encore déterminé pour l'ensemble des stations par une discrétisation du temps. Cette discrétisation suppose une synchronisation des stations sur une horloge commune, ce qui pose des difficultés. Si les transmissions de deux messages, ou plus, ont lieu simultanément, ces messages sont perdus; on dit qu'ils sont entrés en collision [34].

Les limites et l'efficacité de cette technique ont été testées par une simulation statistique d'un bus auquel sont reliés quatre capteurs émetteurs de données à l'Annexe 1. Selon la méthodologie suivante, en tout temps un capteur peut émettre sur le bus. Lorsqu'une collision se produit, l'émission pour le cycle de simulation est annulée et les données à émettre sont placées dans des files d'attente. La réémission de ces données pourra être tentée après

un délai aléatoire attribué à chacun des capteurs du réseau. Donc à chacun des intervalles de temps les opérations suivantes sont effectuées:

- On génère des nouvelles données pour chaque capteur, pour cela on génère un nombre pseudo-aléatoire par capteur et on vérifie si le nombre ainsi généré se trouve dans l'intervalle défini entre 0 et la valeur maximum de l'intervalle aléatoire (i.e. cette valeur est égale à la probabilité d'émission du capteur multiplié par 2^{16} -1 (mot de 16 bits)). Si aucun des nombres générés n'entre dans l'intervalle associé au capteur, alors il n'y a pas de nouvelles données qui vont être générées à ce cycle de la simulation. On s'appuie ici sur le fait que la génération de nombres pseudo-aléatoires posséde une distribution uniforme.

- On vérifie pour chaque capteur s'il a des données à émettre. Si oui, alors il faut le mettre dans la file d'attente respective, vérifier si c'est la file maximum et augmenter le nombre total des données à émettre.

- On vérifie s'il y a une collision (c'est-à-dire, si deux capteurs ou plus émettent en même temps). Si oui, alors il faut incrémenter le nombre d'intervalles avec collisions, augmenter le nombre total de collisions avec le nombre de collisions à cet intervalle et tirer au hasard, pour chaque capteur impliqué, un nombre pseudo-aléatoire entre 1 et délai maximum. Ce dernier correspond à la valeur maximale, en nombre de cycles, du délai qu'un capteur peut éventuellement attendre avant de tenter une réémission. S'il n'y a pas de collision, alors le capteur autorisé, qui a des données à émettre, l'envoie (i.e. on l'enlève de la file d'attente).

De part les résultats obtenus, il semble que le délai de réémission a un effet direct sur le nombre de collisions, l'efficacité et la file d'attente des capteurs (Fig 2.7):

- Si le délai est trop grand, les capteurs peuvent attendre longtemps sans avoir l'autorisation d'émettre; ce qui aura pour effet d'augmenter les files d'attente (mémoire) des capteurs (Fig 2.7.a), de diminuer le nombre de collisions (Fig 2.7.b) et de provoquer un rapport efficacité (Fig 2.7.c) élevé, étant donné le petit nombre de collisions. La raison est bien simple: dû au fait que l'intervalle de temps dans lequel un capteur peut émettre a des chances d'être beaucoup plus loin, alors il laisse la chance aux autres capteurs de pouvoir émettre (donc moins de collisions).

- Si le délai est trop petit, ceci aura pour effet d'augmenter les files d'attente (Fig 2.7.a), d'augmenter le nombre de collision (Fig 2.7.b) et d'avoir une efficacité (P/(P+C) avec P étant la valeur calculée en retirant les données toujours en attente à la fin de la simulation, et C le nombre de collision) peu élevée (Fig 2.7.c), comme le délai est petit alors les capteurs sont plus souvent autorisés à émettre, il y a donc un plus grand risque de collision.

- Si le délai est idéal, il est difficile de trouver un délai optimal, car il dépend de la probabilité de chaque capteur d'avoir des données à émettre. Plus il y a des capteurs qui émettent beaucoup, plus le délai devra être élevé.

Afin d'éviter toutes les procédures de retransmission des données perdues, causées par des collisions, par le choix d'un délai de réémission optimal et par l'ajout des tampons pour les nouvelles données générées, on va

utiliser la technique de sélection avec le contrôle centralisé (oubien la stratégie de communication par invitation à émettre et à recevoir), pour l'échange d'information entre le micro-ordinateur et les capteurs intelligents connnectés au réseau. Cette technique permet au micro-ordinateur (le maître) de gérer l'ensemble des capteurs connectés au bus. Ces capteurs sont interrogés à tour de rôle, par le maître. Une fois qu'un capteur termine sa transmission, il rend le contrôle au maître qui va interroger les autres capteurs (esclaves).

II-2-1-2- Stratégie de communication par invitation à émettre et à recevoir

Dans le cas d'une liaison multipoint, l'exploitation est, dans la quasi-totalité des cas, à la charge de la station centrale qui communique tour à tour avec les autres stations; la station qui reconnait son adresse accepte le message émis en scrutation ou émet son information en sélection. La procédure de transmission du central gère simultanément autant de liaisons point-à-point que des stations connectées sur le canal, tandis que celle d'une station, après avoir été sélectionnée se contente de gérer une seule procédure point-à-point.

La façon la plus simple de gérer une liaison point-à-point consiste à introduire une structure hiéarchique en assignant à une des stations la responsabilité de gérer la liaison. Cette station est appelée primaire ou pilote. L'autre station, qui est appelée secondaire ou tributaire, exécute les commandes qu'elle reçoit du primaire et envoie des réponses. La station primaire assure l'allocation de la voie et la supervision de la liaison, ainsi que la reprise en cas de défaut.

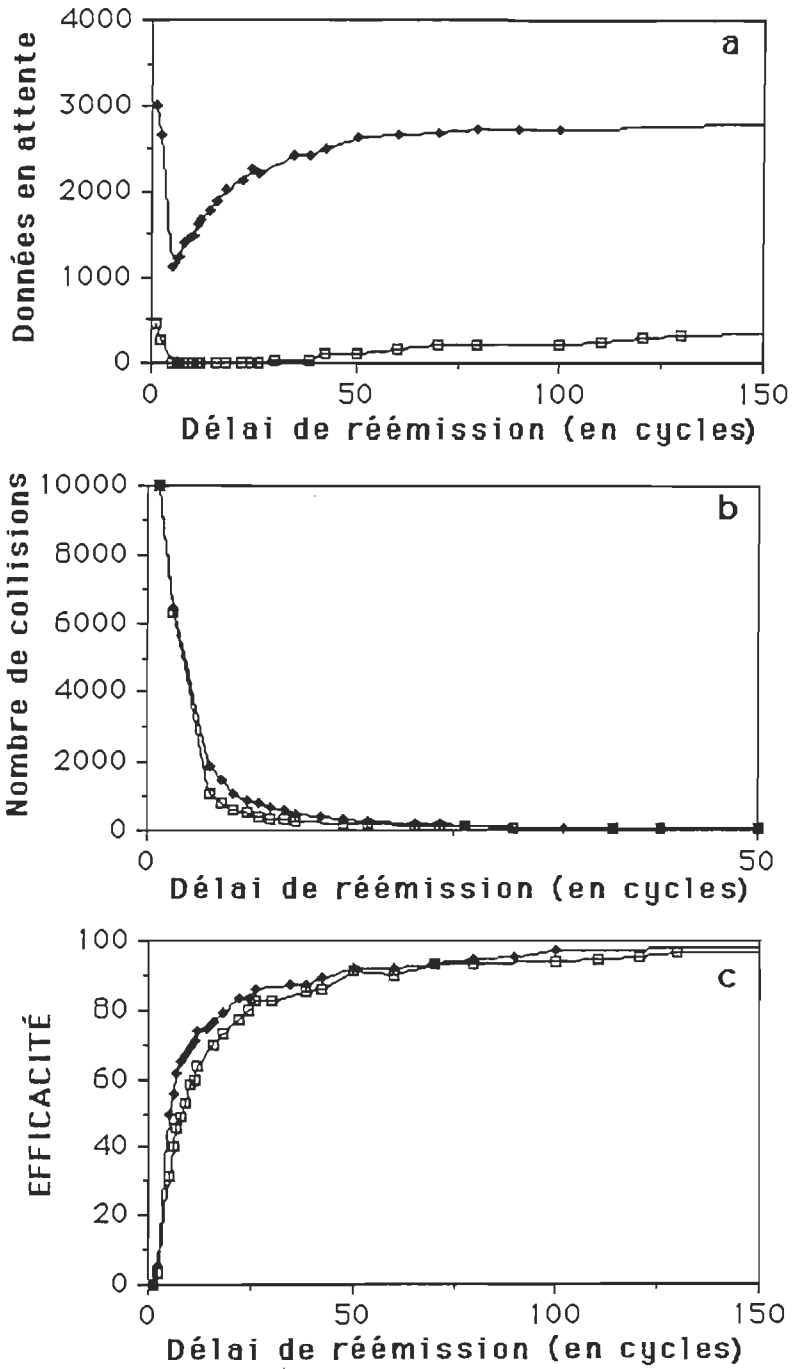


Fig 2.7. Résultats de simulation en bus sur lequel quatre capteurs sont connectés

◆◆◆◆ Probabilité d'émission d'un capteur = 30%

□□□□ Probabilité d'émission d'un capteur = 05%

Les messages échangés sur la liaison peuvent contenir soit de l'information utile appelée message ou information, soit des commandes ou des données de supervision. En mode de commande, la liaison est donc exploitée par l'envoi de commande du primaire vers le secondaire. Ce dernier réagit à ces commandes en retournant des réponses (Fig 2.8) . Les états primaires et secondaires sont permanents, et les stations primaires et secondaires sont généralement réalisées de façon différente. Une station secondaire est en principe plus simple qu'une station primaire, puisqu'elle n'a pas de fonctions de gestion de la liaison. Une station qui n'est à un instant donné ni émettrice ni réceptrice de message se trouve dans un état temporaire neutre [39].

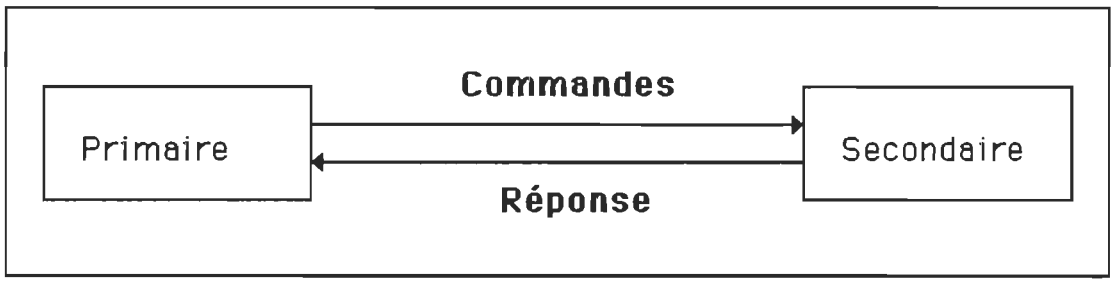


Fig 2.8 . **Liaison de données en mode de commande.**

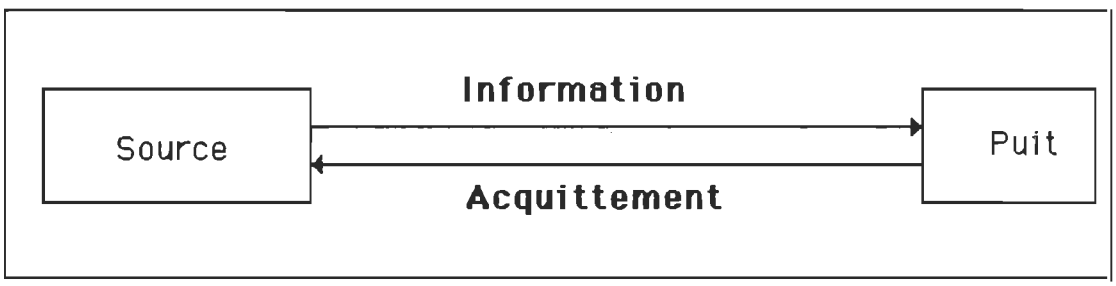


Fig 2.9 . **Source et puits d'informations.**

Par convention, l'information s'écoule toujours d'une source vers un puits (Fig 2.9) , et les accusés de réception circulent toujours d'un puits vers une source. Avec une stratégie point à point hiérarchique, la liaison peut fonctionner sous le contrôle de la station primaire, soit par invitation à émettre, soit par invitation à recevoir. Lorsque le contrôle est de type invitation à recevoire ou sélection, le primaire, qui est associé à une source d'information, prend l'initiative de transférer le message vers le puits après être assuré que le secondaire est prêt à recevoir (Fig 2.10). Le contrôle de la liaison est de type invitation à émettre lorsque la station primaire est associée à un puits et la station secondaire est associée à une source. Dans ce cas, le secondaire transfère le message vers le primaire après réception d'une invitation à émettre en provenance de ce dernier (Fig 2.11).

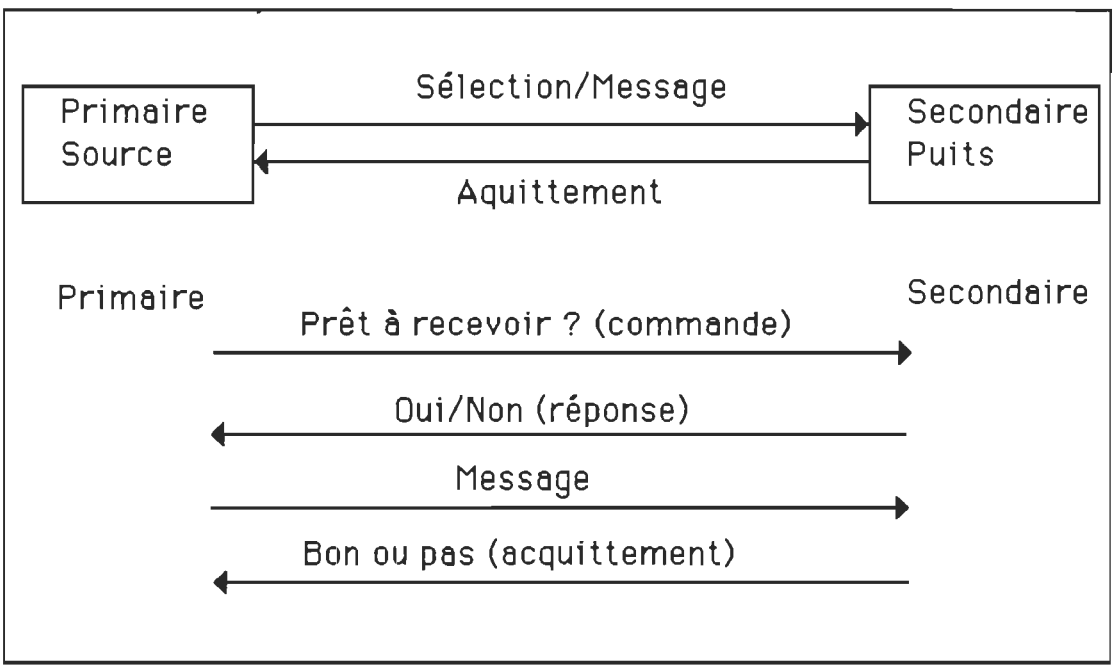


Fig 2.10. **Contrôle d'une liaison point à point par sélection**

Les liaisons de données à contrôle hiérarchique peuvent assurer des transmissions bidirectionnelles de message en combinant deux stations unidirectionnelles de sens opposés. La configuration symétrique non équilibrée, obtenue en superposant deux stations unidirectionnelles de sens opposés de type invitation à recevoir, est montré sur la Fig 2.13. Chaque station comporte alors une fonction primaire associée à une source et une fonction secondaire associée à un puits. Notons ici que la configuration symétrique aurait pu également être réalisée en associant deux stations unidirectionnelles de type invitation à émettre.

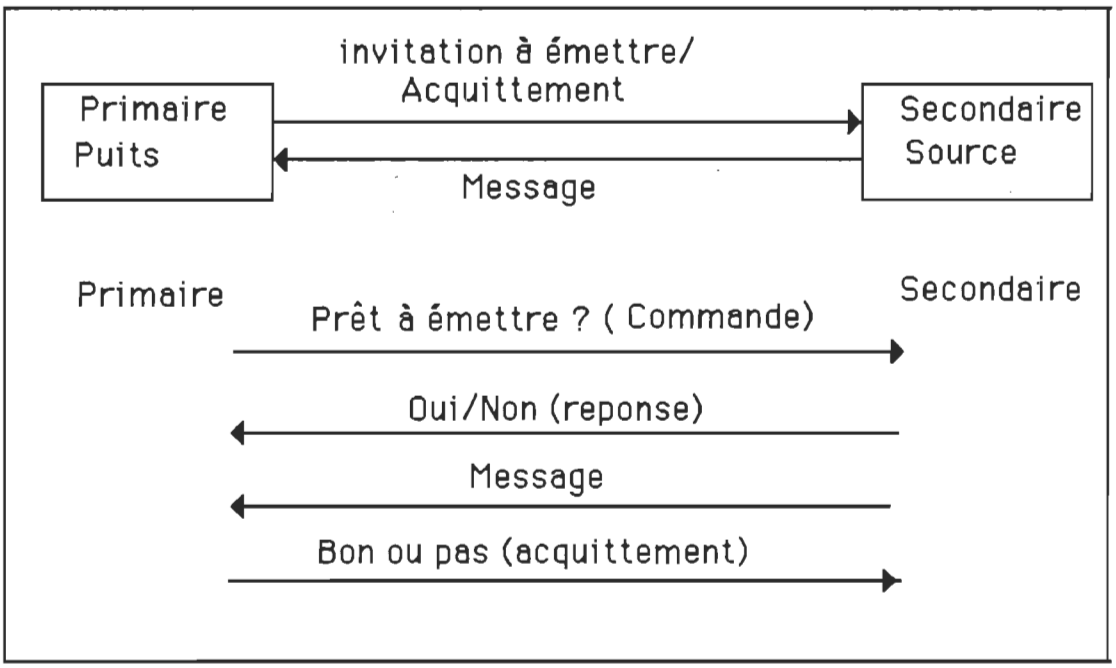


Fig 2.11 **Contrôle d'une liaison point-à-point par invitation à émettre**

Ce dernier type de liaison n'est en pratique guère utilisé, car la commande d'invitation à émettre n'est suivie d'un transfert de message que si

le secondaire dispose d'information à envoyer à cet instant. La procédure invitation à émettre expoite donc la capacité de la ligne d'une façon moins efficace que la procédure par invitation à recevoir. Notons toutefois que ce type de liaison n'est pas compatible avec le fonctionnement multipoint habituel où le central détient souvent, de façon permanente, la fonction primaire [39,40]. Par contre, la configuration point à point asymétrique est la plus simple, où une des stations est primaire pour les deux sens de transmission (Fig 2.13) . Cette configuration superpose une liaison unidirectionnelle par sélection dans le sens primaire vers secondaire et une liaison unidirectionnelle par invitation à émettre dans le sens secondaire vers primaire. Cette configuration est très utilisée car elle concentre dans une seule station.

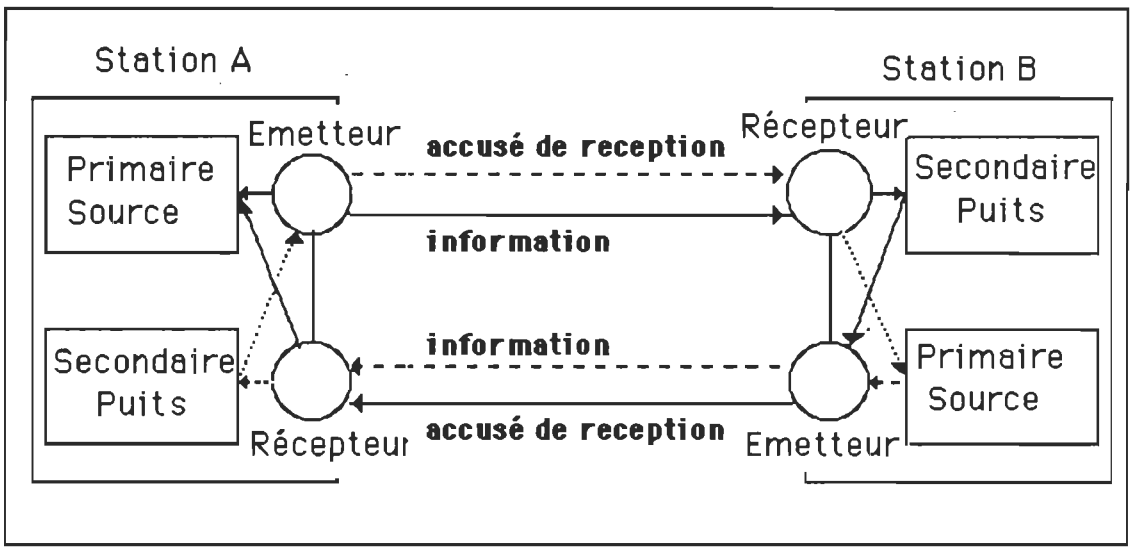


Fig 2.12 **Liaison bidirectionnelle symétrique**

Les fonctions primaires qui sont relativement complexes permettent donc de simplifier la conception des stations secondaires. De plus, cette configuration présente l'avantage d'être compatible avec une liaison

multipoint. Ce type de stratégie est parfois appelé mode "**Maître-Esclave**". En effet, on forme une configuration multipoint à partir de la (Fig 2.13); il suffit de connecter des stations secondaires en parallèle à la station primaire.

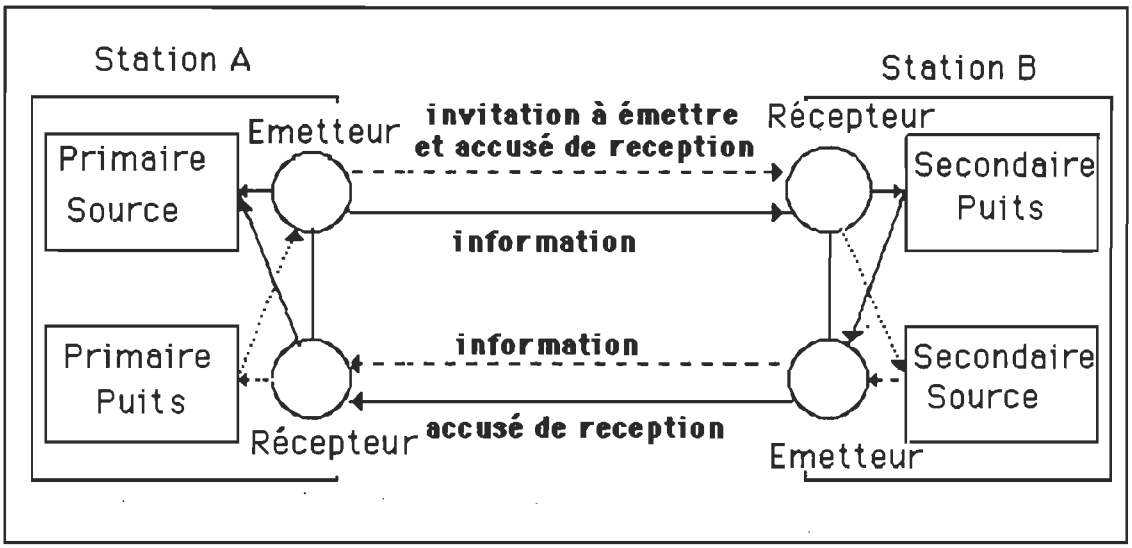


Fig 2.13. **Liaison bidirectionnelle dissymétrique**

II-3-Protocole pour le lien de communication

Le terme protocole est utilisé dans de nombreux domaines et peut être interprété de bien des façons. Une des définitions qui s'apparente le mieux dans ce contexte est la suivante:

«... un protocole est un ensemble formel de conventions définissant le format et le séquencement de l'échange entre deux points [45] ».

L'élaboration du protocole se fera selon les critères suivants:

- Aptitude à établir efficacement une communication entre deux points;

- Élaboration d'un format précis pour les messages;

- Minimisation du transfert de l'information redondante;

- Minimisation de la complexité des points d'accès.

II-3-1-Analyse du format utilisé pour les messages

Les informations qui sont transmises sur une liaison série peuvent l'être dans deux modes de trasmission, asynchrone ou synchrone. Dans le mode de transmission synchrone, le transfert des informations se fait sous forme de trame, tous les bits à la suite les uns des autres sans espace (Fig 2.14). Au niveau du format des messages, la plupart des développements récents ont eu lieu dans le domaine des réseaux locaux. À titre d'exemple, le protocole HDLC "High-level Data Link Control" sera analysé. Il est intéressant de noter que la norme X-25 qui est de plus en plus utilisée lors de la transmission par paquets via les lignes téléphoniques, utilise le protocole HDLC. Il faut noter que le format HDLC a été prévu pour être utilisé avec un champ d'information très grand, ce qui n'est pas le cas du système à l'étude.

Il faut se tourner vers les systèmes commerciaux plus modestes afin de trouver des exemples de formats plus adéquats. On remarque, entre autres, les systèmes développés par les compagnies N.V.Philips (Format I^2C) [34] et Intersil (Format Remdac) [27]. Ces deux systèmes utilisent une structure orientée en fonction des bits plutôt qu'en fonction d'octets comme c'est le cas dans le réseaux locaux. Dans le cadre d'une application où le nombre de bits d'information est peu élevé, ces deux types de format sont nettement plus performants.

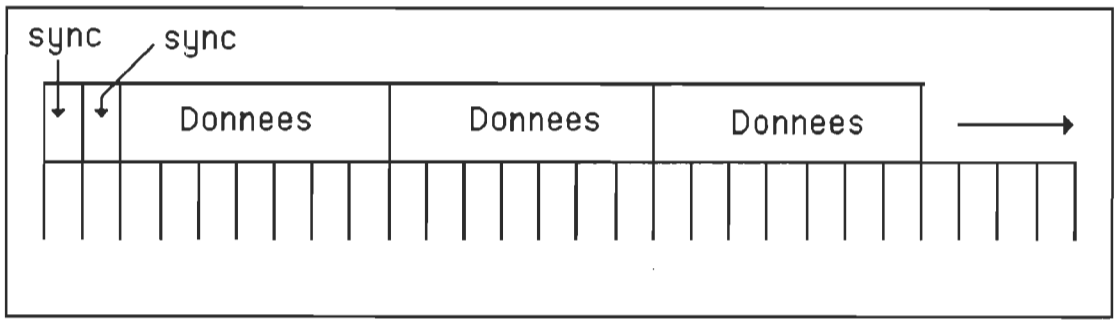


Fig 2.14. **Mode synchrone**

Ces exemples serviront à orienter le travail dans la définition d'un format de transmission spécifique pour notre application. D'autre part, la carte de communication qui existe dans le micro-ordinateur utilisé est de type asynchrone.

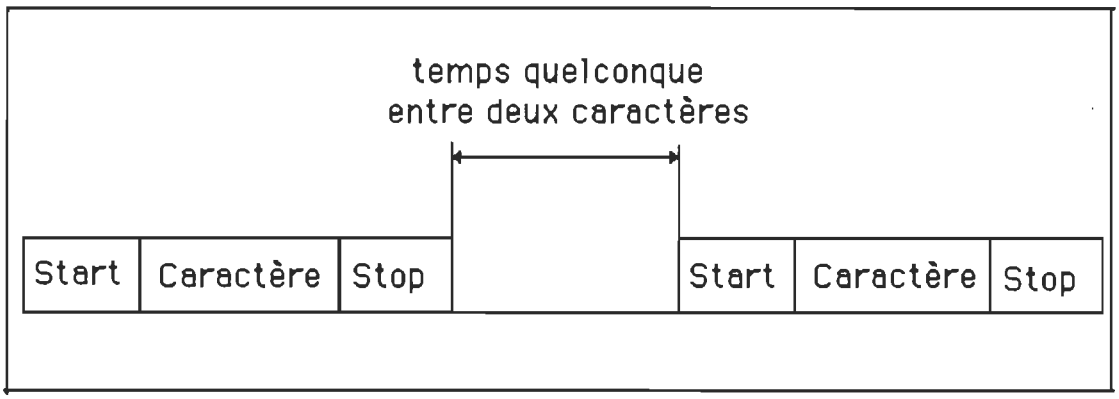


Fig 2.15. **Mode asynchrone.**

En mode de transmission asynchrone, les informations qui sont codées sous forme de caractères peuvent être émises à tous moments (Fig 2.15). Dans ce mode de transmission chaque caractère est encadré par des éléments délimiteurs: un élément de début, appelé START, et un élément de fin appelé STOP, dont les durées sont supérieures à celle d'un moment élémentaire. Pour cela la transmission est parfois appelée transmission START-STOP. L'information redondante d'encadrement START-STOP est importante au point de vue de l'information utile transmise.

II-3-2- Format des messages utilisé

Pour reconnaître les données groupées en blocs, on doit définir des signes de reconnaissance de début et de fin de ces groupements. On utilise, dans un code donné, des caractères spéciaux dits caractères de commande, définis spécialement pour indiquer le début et la fin du bloc de données. L'utilisation des codes spéciaux permet d'éviter que ces signes d'encadrement

apparaissent dans l'information à transmettre. Ce phénomène étant possible il faut prévoir un mecanisme pour s'assurer que la procédure de commande ne reconnaîsse que les vrais caractères de commande, c'est-à-dire pour s'assurer que la liaison de données soit indépendante des codes ou alphabets utilisés, (notre cas le code ASCII). Ce code se divise en deux groupes, les caractères de contrôle non imprimables et les caractères imprimables. L'utilisation des caractères de contrôle est très variée on peut citer principalement, les caractères de gestion de flux logicielle.

En principe, si l'échange de l'information se fait seulement entre deux points fixes (point à point), il n'y a pas de besoin d'identification ni pour l'origine, ni pour la destination, puisque l'une est unique vis-à-vis de l'autre. Mais une liaison de données peut comporter plus de deux stations (multipoint). Dans ce cas, il est nécessaire d'identifier l'origine et la destination pour les échanges de données. Si l'échange se fait toujours entre une station centrale et une station quelconque de la liaison, il suffit d'identifier à chaque échange la deuxième station.

D'une manière générale, au niveau da la liaison, on donne à chaque station une adresse (ou nunéro), représentée par un ou plusieurs caractères, ou une séquence d'éléments binaires, qui l'identifie.

Alors, le protocole choisi consiste à ce que l'ordinateur central envoie aux capteurs intelligents, raccordés sur la ligne, l'adresse du capteur qui est représenté pas un caractère (chiffre) en ASCII.

En recevant cette commande (adresse), le capteur selectionné sur la ligne envoie automatiquement son message enregistré dans sa mémoire vers l'ordinateur central sous le format suivant:

STYRENE	**TEMPERATURE**	**PRESSION**	**FIN**

Fig 2.16 Format du message émis par les capteurs intelligents

Styrène (c'est à dire la concentration), température et pression relative sont représentés par 15 caractères (5 caractères pour chaque donnée) et 2 caractères pour la fin du message. Une fois que l'ordinateur reçoit ce message, il va décomposer ce dernier en trois données significatives pour le traitement.

CHAPITRE III

MATERIEL UTILISE ET SON OPERATION

Dans le chapitre précédent, on a choisi la transmission série asynchrone, sur une ligne bifilaire, comme moyen de communication. Donc pour réaliser la conception des circuits de ce type de transmission on doit utiliser deux types fondamentaux de circuits intégrés. Le premier type est le convertisseur de niveau qui transforme les signaux TTL en niveaux RS-232C, RS-423A ou RS-422A, le second type de circuit est l'émetteur/récepteur.

Les émetteurs/récepteurs effectuent la conversion parallèle-série nécessaire pour émettre, et la conversion série-parallèle nécessaire pour recevoir un flux de bits série. Avant que les émetteurs/récepteurs n'existent en circuits intégrés, on utilisait des registres à décalage pour effectuer la même fonction. Les émetteurs/récepteurs récents et plus perfectionnés fournissent des performances accrues pour un prix analogue. Certains des émetteurs/récepteurs les plus récents ont des générateurs de fréquence de bits internes [41].

On utilise dans cette étude, deux types d'émetteurs/ récepteurs, l'un pour la station centrale de contrôle et de gestion (micro-ordinateur), l'autre

pour les capteurs intelligents (microcontrôleur M68HC11 - voir Annexe 4).

III-1- Carte de communication série [41].

Un micro-ordinateur (80286) compatible d'IBM est choisi comme station centrale de contrôle pour le réseau de capteurs intelligents. Ce micro-ordinateur dispose d'un port série, ce dernier a été conçu a l'aide d'une carte de communication aysnchrone. Le boitier d'entrée/sortie série programmable trouvé dans cette carte de communication est l'adaptateur de communication asynchrone (ACE) 8250 de National Semiconductor. L'ACE-8250 a été conçu pour des capacités étendues; non seulement il effectue les conversions série-parallèle et parallèle-série pour les transmissions asynchrones, mais il comporte aussi un générateur de fréquence de bit et une structure d'interruption complexe.

La figure 3.1 représente chacun des composants essentiels d'une unité de communication « asynchrone » . Le mot asynchrone signifie que les différents signaux transitant entre les unités de communication arrivent généralement au hasard et non au rythme d' une horloge principale . Dans les systèmes de communication entre systèmes à microprocesseurs, la conversion de données du mode parallèle en mode série est généralement assurée par un boîtier appelé un UART (ang. Universal Asynchronous Receiver/Transmitter). C'est l'UART qui se charge également de l'adjonction de bits supplémentaires à chaque octet transmis et qui fixe la fréquence de transmission du signal, de sorte que celui-ci soit transmis avec la vitesse de transmission adéquate. Cette

vitesse de transmission est, pour ce qui nous intéresse, mesurée en bits par seconde (bauds) et elle correspond à la vitesse à laquelle chaque octet et les bits supplémentaires qui lui sont attachés sont transmis. Bien que chaque caractère (chaque octet en l'occurrence) soit transmis et reçu par l'UART de façon asynchrone , chacun des bits qui le composent doit être transmis et reçu à la vitesse fixée, sans quoi l'UART s'y perdrait complètement. Nous allons examiner étape par étape le proccessus de transmission d'un octet vers la périphérie du micro-ordinateur.

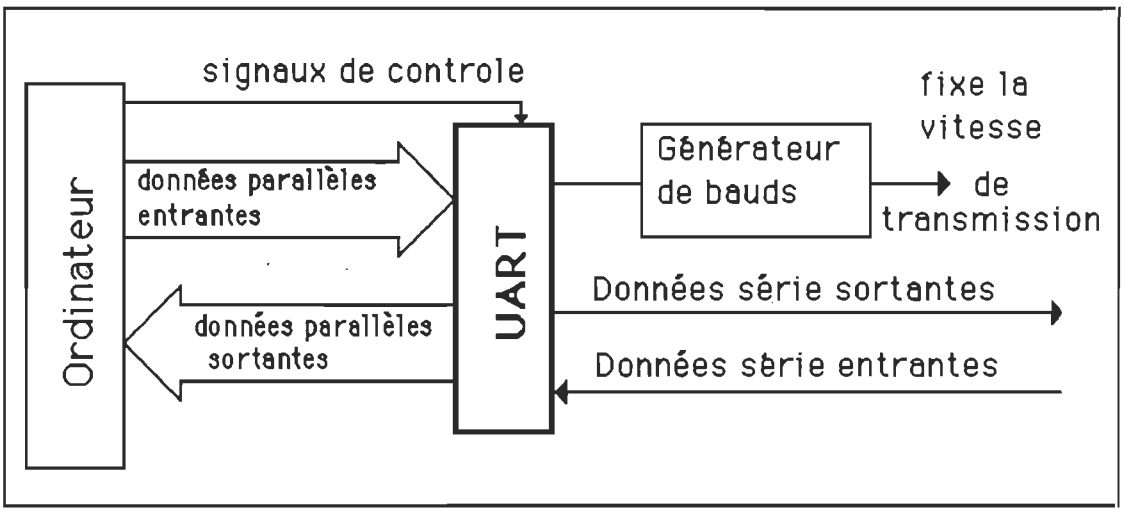


Fig 3.1 . **Unité de communication asynchrone.**

III-1-1-Formatage des données

A chaque extrémité de la ligne de communication, les ordinateurs sont généralement dans ce que l'on appelle un état de repos vis-à-vis de la transmission ou de la réception de caractères. Dans cet état de repos, on transmet une tension continue équivalente à un "1" logique et encore appelée

«mark» en anglais. C'est un procédé de sécurité. En effet un niveau bas continu "0" n'est alors possible que dans une configuration où l'ordinateur est inopérant. Un "0" logique encore appelé «space» en anglais, se prolongeant pendant une bonne fraction de seconde ou plus est utilisé comme signal d'arrêt par de nombreux systèmes et correspond à des interruptions ou à des initialisations. La transmission d'un nouveau caractère est toujours signalée par un bit de départ qui est toujours une interruption de la transmission de tension (d'où un «0» logique). Ce bit de départ est suivi de 5 à 8 bits de données, le nombre exact dépendant du codage utilisé pour la transmission . Le séquencement de la transmission est organisé de telle façon que le bit de poids le plus faible (LSB) soit transmis le premier, suivi des bits de poids croissant. Il est possible qu'un bit de parité succède aux bits de données. Ce dernier sert à s'assurer de la validité de la transmission, sa valeur est 0 ou 1 selon la convention de parité utilisée par les ordinateurs, et il est fonction des bits composant la donnée à transmettre. Par exemple, si le signal est transmis avec une convention de parité impaire alors le nombre de bits (de données et de parité) à «1» doit être impair. Par exemple, si l'on transmet le mot 11001011 avec une parité impaire, alors la valeur du bit de parité est 0 car la donnée contient un nombre impair de 1. Inversement si le même mot 11001011 était transmis avec une parité paire, le bit de parité prendrait alors la valeur 1 de façon à rendre pair le nombre de 1 de données plus le bit de parité.

Le bit de parité, s'il existe, est suivi d'un ou de plusieurs bits d'arrêt qui sont toujours des «1». Jusqu'à présent trois options ont été offertes: 1, 1,5 et 2 bits d'arrêt. Cependant, la plupart des UART n'offrent que deux de ces trois options: 1 ou 2 bits d'arrêt car le 1,5 bit (au sens de l'horloge de l'UART) n'est

utilisé que dans des configurations de systèmes Télétype à faible vitesse qui utilisent généralement des méthodes électromécaniques de génération et de réception de signaux. Dans tous les cas cependant, le nombre choisi de bits d'arrêt dépend du protocole de transmission choisi. Une fois cette séquence de bits transmise, le système retourne à l'état de repos jusqu'à ce qu'un prochain caractère soit transmis.

III-1-2- Fonctions de l'émetteur.

Le micro-ordinateur communique à l'UART les signaux de contrôle nécessaires à la fixation des paramètres usuels; vitesse de transmission, nombre de bits d'arrêt et convention de parité utilisée. Ces signaux arrivent sur les registres de contrôle de l'UART et restent valides jusqu'à ce que des signaux émanant du micro-ordinateur les altèrent. La transmission des signaux de contrôle étant efectuée, l'UART est prêt à traiter des données. L'ordinateur indique à l'UART qu'un octet est sur le point d'être envoyé en validant l'adresse périphérique correspondant à l'UART. Le circuit de validation est lui aussi un circuit externe à l'UART. L'UART est représenté avec 8 bits de données arrivant en parallèle (Annexe 5). L'octet de données arrive sur le registre de données de l'émetteur puis les données transitent vers le registre de données de l'émetteur sous le contrôle d'une horloge de synchronisation. En chemin, l'octet peut être modifié par adjonction de données supplémentaires fournies par le générateur de bit de parité et le sélecteur de longueur de mot.

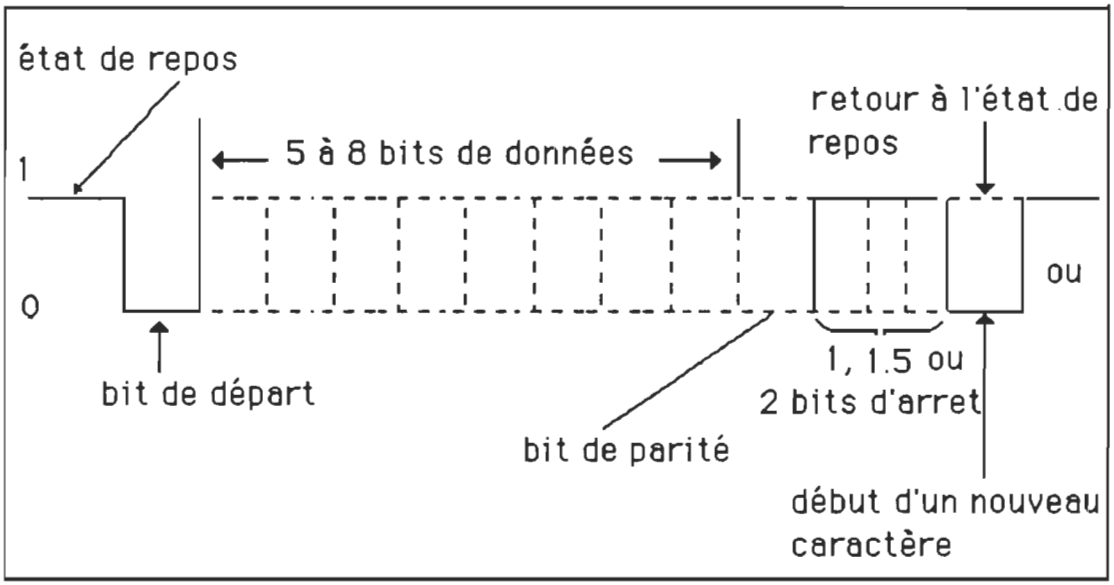


Fig 3.2. **Formatage des données**

Selon que la parité sélectionnée par l'ordinateur est paire ou impaire, le générateur de bit de parité accorde ce bit en fonction du nombre de "1" de l'octet. Il peut être aussi envisagé de ne pas adjoindre de parité et dans ce cas aucun bit de parité n'est généré.

La sélection du nombre de bits d'arrêt complète la tâche préparatoire du registre de l'émetteur. A la période d'horloge suivante, celui-ci envoie le caractère reconstitué au rythme d'un bit par période d'horloge. La durée d'un intervalle entre chaque bit est déterminée par la vitesse de transmission. Par exemple, dans une transmission à 300 bauds , un caractère de 10 bits (un bit de départ, sept bits de données, un bit de parité, et un bit d'arrêt) est transmis en 1/30e de seconde (10 bits avec une durée de 1/300e de seconde par bit). Comme dans le cas de la réception de données en entrée, la plupart des systèmes de communication possèdent des circuits registre additionnels intercalés entre la

sortie de l'UART et la ligne de communication extérieure pour s'affranchir des éventuelles différences de tension entre les deux équipements.

III-1-3- Fonctions du récepteur

Comme le mode de transmission est asynchrone, le récepteur ne sait pas vraiment à quel moment un caractère doit lui parvenir via la ligne de communication. Il lui faut donc d'abord être en mesure de faire la distinction entre le bruit et le signal, car, dans la réalité, il est courant que la ligne de communication soit bruitée. Cette distinction est réalisée en échantillonnant la ligne d'interface à la ligne de communication à une vitesse 16 fois supérieure à celle où les bits attendus arriveraient. Ainsi pour une transmission à 300 bits/s, le récepteur de l'UART échantillonnerait les informations provenant de la ligne 4 800 fois/s . Lorsqu'un "0" est détecté, un comptage est déclenché sur les 8 intervalles d'échantillonnage suivants (à savoir les 8 prochaines 4800 centièmes de seconde, ou encore un intervalle d'un demi-bit). Le récepteur vérifie alors si la ligne est encore au niveau "0" ; si c'est le cas, on estime que le signal présent est de l'information et non du bruit . Cette estimation est justifiée par la théorie qui indique qu'il est relativement improbable (quoique non impossible) que les signaux de bruit soient à «0» précisément et simultanément, au début et à la moitié, de l'intervalle de temps correspondant à un bit.

La plupart des autres fonctions du récepteur sont exactement inverses de celle de l'émetteur. L'horloge et le registre à décalage de données

décalent les bits arrivant par la ligne, un par un, dans le registre du récepteur. Durant ce processus, la longueur et la parité du mot sont vérifiées, et si sa parité ne correspond pas à celle attendue par le récepteur un signal d'erreur est généré. Lors du test de la longueur du mot, on s'assure que le(s) dernier(s) bit(s) du caractère est (sont) à «1». Si tel n'est pas le cas, un signal d'erreur de forme (framing error signal) est généré pour indiquer une anomalie relative au caractère. Si le système est bien régulé, les erreurs de forme sont rares. En effet, la cause principale d'erreurs de forme est l'utilisation d'un récepteur et d'un émetteur distants fonctionnant à des vitesses de transmission différentes. Par exemple, si le récepteur attend un signal à 600 bauds et que le signal effectivement présent est à 300 bauds, le récepteur considérera les 5 premiers bits comme 2 x 5 = 10 bits, sans attacher d'importance au fait que la donnée alors reçue soit constituée de paires de bits semblables. Il arrivera alors que le dixième bit soit un «0» et le récepteur signalera une erreur de forme (se reporter à la définition du code 10 bits utilisé plus haut).

Un autre type d'erreur existe également: il s'agit d'une erreur de débordement (overrun error). Cette dernière est causée par l'arrivée du caractère suivant, avant que le récepteur ait eu le temps de procéder complètement à l'envoi du précédent vers l'ordinateur. On diminue les possibilités de débordement en transférant les données à l'ordinateur en parallèle, en un temps correspondant à une fraction de période d'un bit de récepteur, et en envoyant chaque caractère dans un registre dès qu'il est considéré comme complet, au niveau du registre du récepteur.

Avec les vitesses de transmission classiques (de 150 à 1 200 bauds), les

erreurs de débordement sont rares et la plupart des micro-ordinateurs arrivent à supporter de telles vitesses, tant qu'ils opèrent en mode simple tâche , et tant que le programme de contrôle du modem est efficace. Par conséquent, de nombreux micro-ordinateurs ignorent ce signal de débordement provenant de l'UART. Cependant, à des vitesses plus importantes, comme par exemple dans les réseaux directs ou locaux, estimer que l'ordinateur peut traiter parfaitement la communication sans vérification d'éventuelles erreurs de débordement, peut s'avérer dangereux.

Dans tous les cas, qu'il y ait eu erreur ou non, un signal «donnée prête» (data available) est généré pour indiquer à l'ordinateur qu'un caractère est prêt et peut être lu.

Par la suite, si le micro-ordinateur est distrait et qu'il oublie de prendre en compte le signal "donnée prête", l'UART continuera, lui à recevoir des caractères et à les transférer au registre du récepteur, en envoyant également le signal de débordement. Il en résultera la perte d'une grande quantité d'information. Il faut donc qu'on tienne compte de cette éventualité, avant de négliger des signaux d'erreurs.

III-2- Entrées/sorties du capteur intelligent.

III-2-1- Description générale.

Le SCI est une interface de communication série du type UART , qui utilise les signaux non retour-à-zéro (NRZ), et le format suivant: un bit de

départ, huit ou neuf bits de données et un bit d'arrêt.

L'émetteur et le récepteur sont tous les deux des double registres, ainsi, tous les caractères peuvent être manipulés facilement, même si l'unité centrale du processeur est en retard en répondant à l'exécution d'un caractère individuel. L'émetteur et le récepteur fonctionnent indépendamment, mais ils utilisent le même format de données et le même générateur de fréquence.

Le récepteur inclut un nombre de traits avancés pour assurer une réception de données et pour assister au développement d'un réseau de communication efficace. Le microcontrôleur M68HC11 resynchronise l'horloge du bit de réception sur toutes les transitions dans un système de cours de bits plutôt qu'au début seulement du temps de bit de départ; alors, des différences dans le générateur de fréquence entre la machine d'envoi et le CPU ne causent pas aussi souvent des erreurs de réception. Trois échantillons de niveaux logiques sont pris près du milieu de chaque cadence, et la logique majoritaire décide le sens pour le bit. Le récepteur a aussi l'habilité de devenir un mode standby (appelé récepteur WakeUp) pour ignorer des messages pour un autre récepteur. Automatiquement, la logique réveille le récepteur pour qu'il ait assez de temps pour avoir le premier caractère du prochain message. Le WakeUp réduit dramatiquement l'intervention du CPU dans une communication multipoint.

III-2-2- Fonctionnement de l'SCI.

Le coeur du transmetteur est le "Registre à décalage de

transmission". Ce registre de décalage reçoit toujours les données du "Registre de transmission ". Les données entrent dans le registre de transmission quand on écrit logiciellement dans le registre SCDR. Chaque fois que les données sont transférées dans le registre de décalage de transmission, un zéro est chargé dans le bit le moins significatif (LSB) du registre pour agir comme un bit de départ, et un logique est chargé dans le dernier bit pour agir comme un bit d'arrêt.

Le registre du récepteur "Registre à décalage de réception" est activé par le bit RE (ang. Receve Enable) du registre de contrôle (SCCR2). Le bit M du SCCR1 détermine la longueur du mot. Après la détection du bit d'arrêt d'un caractère, la donnée reçue est transférée au registre SCDR, et le bit RDRF du registre d'état est à un. Lorsqu'un caractère est prêt pour être transféré au registre de réception mais que le caractère précédent n'a pa été lu encore, une condition de dépassement se produit. Dans ce cas de dépassement, la donnée n'est pas trasferée et le drapeau d'état de dépassement (OR) est à 1 pour indiquer une erreur.

Le contrôle de procédure de transmission et de réception se fait, par le contrôle des sept bits du registre d'état. Ces bits sont associés au système SCI; certains de ces bits génerent des interruptions matérielles, tandis que d'autres indiquent simplement des erreurs dans la réception du caractère. Une fois fixé, ces bits restent fixés jusqu'à la mise à zéro par le logiciel. Par exemple, pour mettre à zéro le bit TDRE, le logiciel doit lire le SCSR pendant que le TDRE est fixé, et puit écrire au TDR puisqu'ils sont exactement les étapes normales en réponse au TDRE, on n'a pas besoin d'instruction pour mettre à zéro ce bit.

III-3- Convertisseur de niveau.

Les convertisseurs de niveau sont nécessaires parce que , les niveaux TTL que l'on trouve couramment dans les systèmes à micoprocesseurs ne sont pas directement compatibles avec les niveaux de tension exigés par les normes des interfaces séries. Ces convertisseurs de niveaux peuvent être réalisés au moyen de circuits à transistors discrets mais les émetteurs/récepteurs en circuits intégrés sont plus commodes et en général moins chers.

III-3-1- Normes électriques.

La spécification électrique, la plus courante utilisée pour les communications de données série asynchrones, est celle de la norme RS-232C de la Electronic Industrie Association (EIA) Annexe 6. Elle a la limitation de la vitesse de transmission et la limitation de la distance de transmission. Conformément à la norme, les longueurs de câbles sont limitées à 50 m. En outre, la RS-232C ne possède qu'une seule ligne de retour (masse logique), ce qui contribue à la susceptibilité au bruit. La réduction de la vitesse de transmission diminue la génération de bruit, alors que la réduction de la longueur de câble diminue le couplage entre les conducteurs de celui-ci. Les nouvelles spécifications de signal sont la RS-423A et RS-422A Annexe 6. La RS-423 spécifie une ligne unique semblable à RS-232A, et compatible avec la spécification RS-232. Les caractèristiques électriques de ces trois principales normes sont résumées dans le Tableau 3.1. Le choix de la norme, est fait par rapport aux caractéristiques de la ligne bifilaire décrites dans les paragraphes suivants [42,43] .

Tableau 3.1. **Caractèristiques électriques des convertisseurs de niveaux** [36]

Caractéristiques	RS-232C	RS-423	RS-422
Support de transmission	1 fil	1 fil Coaxial	differentiel
Vitesse de transmission	20 KB	300 KB	10 MB
longueur de cable	50 m	500 m	1 Km
tension de sortie max circuit ouvert	± 25 V	±4 < Vo<= 6V	=< 6V dif
tension de sortie circuit fermé	±15 V	±3.6 V	2 V
Résistance de sortie	3K< R_1 < 7K	450Ω 50 Ω coax	≤ 100 Ω
Courant max en C/Co	0.5 A	< 150 mA	< 150 mA
Zône de transition	± 3V	± 0.3V	± 0.3V

III-3-2- Impédance caractéristique.

Si le temps de montée du signal à transmettre est court par rapport au temps de propagation de la ligne, l'émetteur sera chargé par l'impédance caractéristique de la ligne qui est de l'ordre de 100 Ω pour une ligne bifilaire.

Dans ce cas le coefficient de réflexion est égale à:

$$C_r = (Z-Z_0)/(Z+Z_0)$$

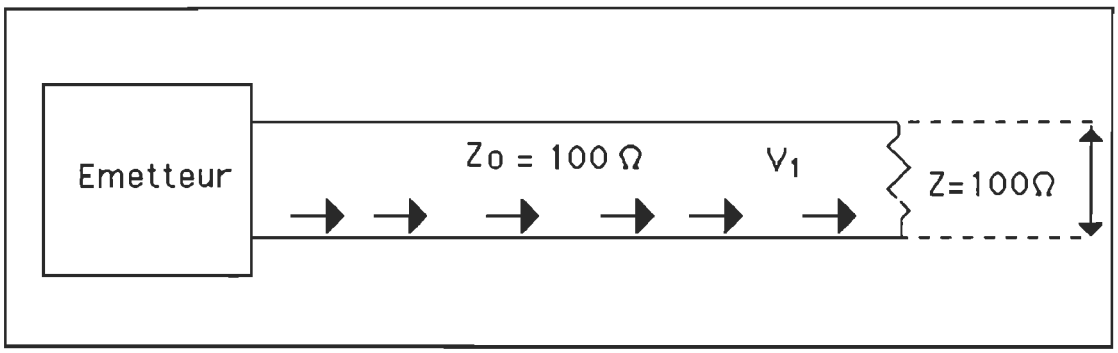


Fig 3.3 . **Ligne bien adaptée**

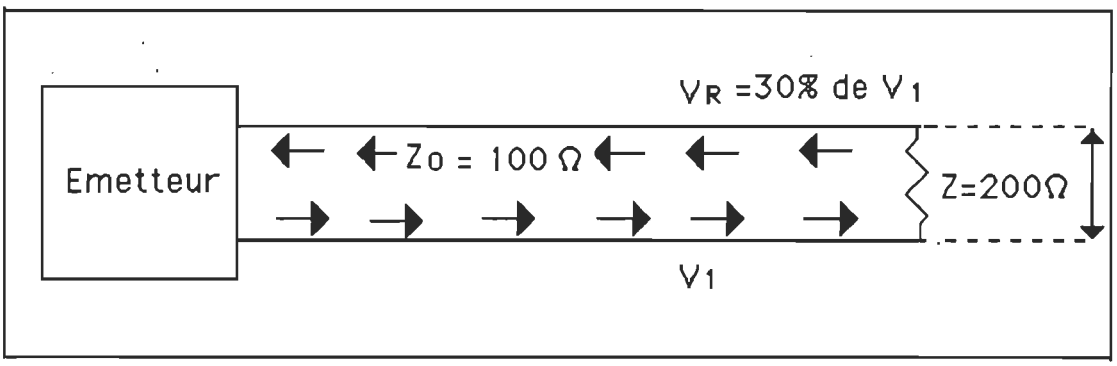


Fig 3.4 . **Ligne non adaptée (Z#Zo : réflexion)**

Si, comme le représente la figure 3.3, Z=Zo, il n'y aura pas de réflexion et la tension disponible au bout de la ligne vaut 100% de la tension incidente V_1. Par contre, la figure 3.4 représente un exemple de mauvaise adaptation: la tension réfléchie V_r égale 30% de la tension incidente et la

transmission est fortement perturbée. En pratique on termine toujours la ligne sur son impédance caractéristique si (**L**[m]/**TH**[m]) >1

M = 1/TH = frequence de modulation en **MHz**

L = longueur de la ligne en **m**

TH = durée de transmission d'un caractère en **s**

(voir figure 3.5)

La vitesse maximum qu'on peut atteindre avec notre protocole asynchrone est 9600 bauds/s. Un caractère sera transmis en (8x1/9600) secondes; donc la fréquence de modulation sera 1200 Hz si la transmission est sur une longueur d'un maximum de 500m

ML = 0.001200 x 500 = 0.6 <1

Dans ce cas, notre transmission ne sera pas perturbée. Alors, la longueur maximale de la ligne dépend surtout de la fréqence de modulation, cela montre qu'il faut avoir un compromis entre la vitesse de transmission et la longueur de la ligne.

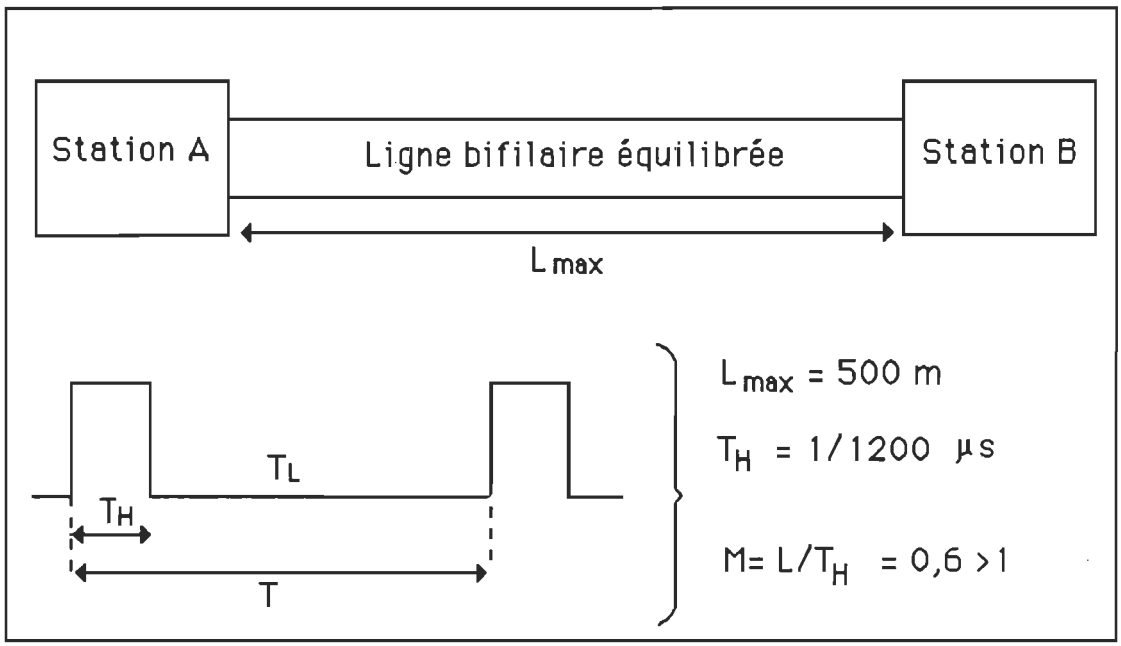


Fig 3.5 **La fréquence de modulation**

III-3-4- Émetteur et récepteur de ligne

On sait que le courant de sortie haut d'une porte TTL est de 0.4 mA (Fig 3.6-a). Selon la loi d'Ohm, dans ce cas le courant de sortie pour un seuil maximum de 3V, on aura un courant de 30 mA (Fig 3.6-b), on ne peut donc pas utiliser un TTL sur une ligne bifilaire adaptée. Pour cette raison les émetteurs de ligne doivent être capables de fournir un courant de l'ordre de 30 mA, or d'après le Tableau 3.1 toutes les normes satisfont cette condition, puisque le courant maximum en court-circuit varie de ±150 mA à ±500 mA. On peut ignorer la résistance d'entrée de plusieurs kΩ des récepteurs de ligne, car si deux résistances sont en parallèle on peut éliminer la plus grande, (Fig 3.7). Pour cette condition on sait que la norme RS-422A peut fonctionner

en haute impédance, donc on peut l'utiliser comme récepteur de ligne.

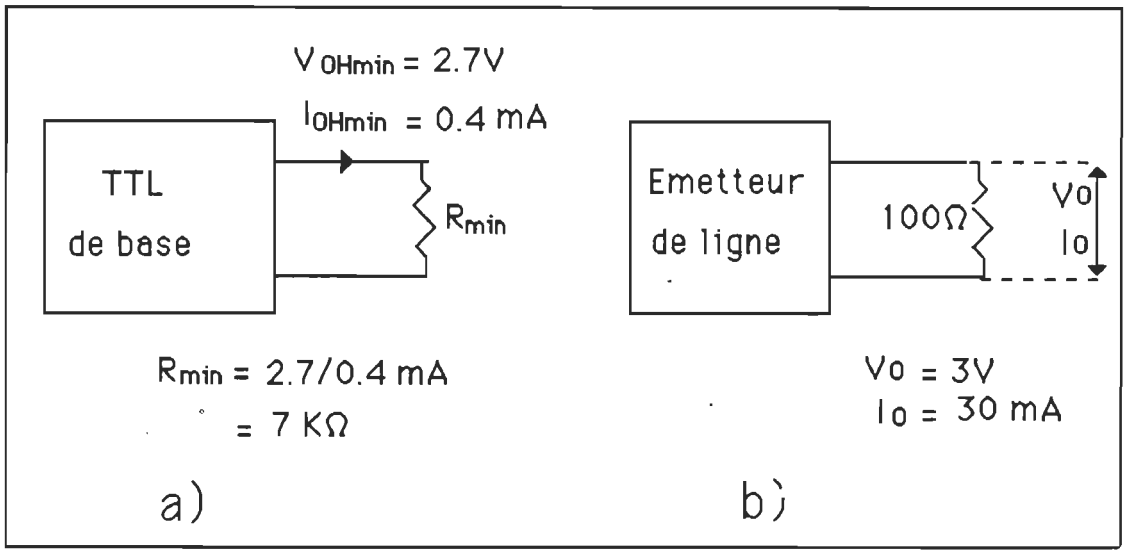


Fig 3.6 **Emetteur de ligne**

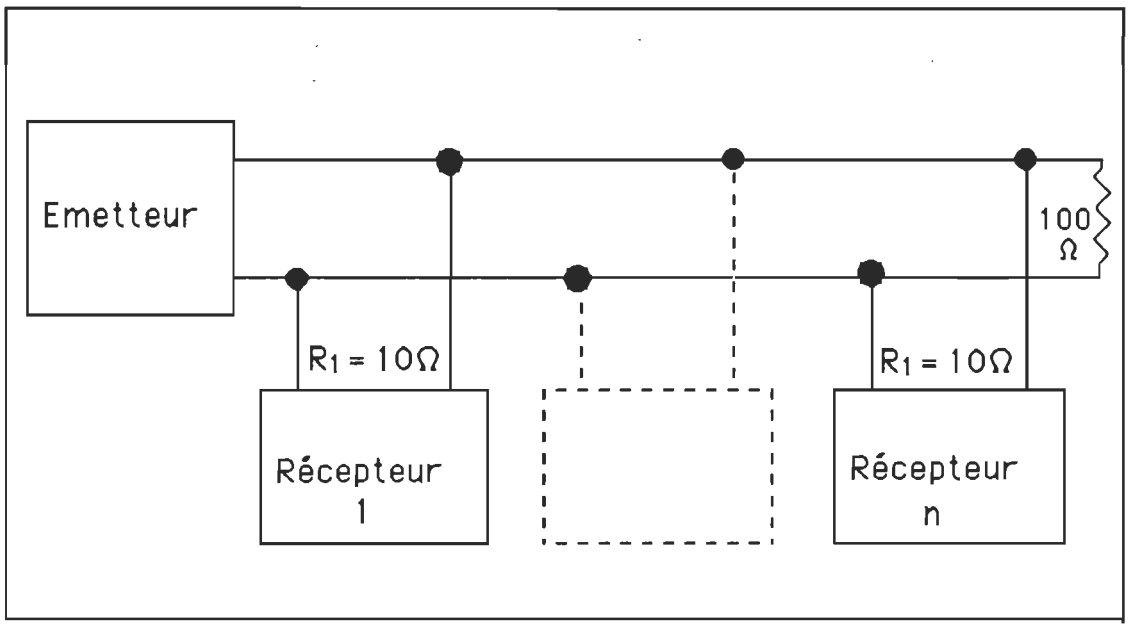


Fig 3.7 **Récepteur de ligne**

III-3-5- Émetteurs et récepteurs différentiels.

Une ligne différentielle est une ligne où l'on utilise deux fils pour envoyer un signal. Cette technique améliore notament le signal parce que la valeur crête-à-crête est, en fait, doublée . Lorsque l'une des sorties de l'émetteur différentiel est à +12 volts, l'autre est à -12 volts, si bien que la difference entre les deux sorties est 24 volts par rapport à la borne de sortie inférieure. Si le niveau logique à l'entrée de l'émetteur change, la borne de sortie qui était à +12volts passe à -12 volts, tandis que l'autre passe de -12 à +12 volts la difference entre les deux bornes est maintenant de -24 volts toujours par rapport à la borne inférieur de l'émetteur. La RS-422 possède un mode pour les données différentielles; ce mode permet une bonne immunité au bruit.

Les résistances terminales de ligne à utiliser avec un émetteur et un récepteur différentiels doivent être dédoublées comme le montre la (Fig 3.8) qui illustre aussi comment l'entrée du récepteur ignore un bruit commun I_z. De plus, le courant dans le fil de masse étant négligeable, la ligne équilibrée ne génère aucun bruit. Il en va tout autrement pour la ligne déséquilibrée représentée à la (Fig 3.9) . Le courant dans le fil de masse est une source de bruit commun avec les circuits voisins.

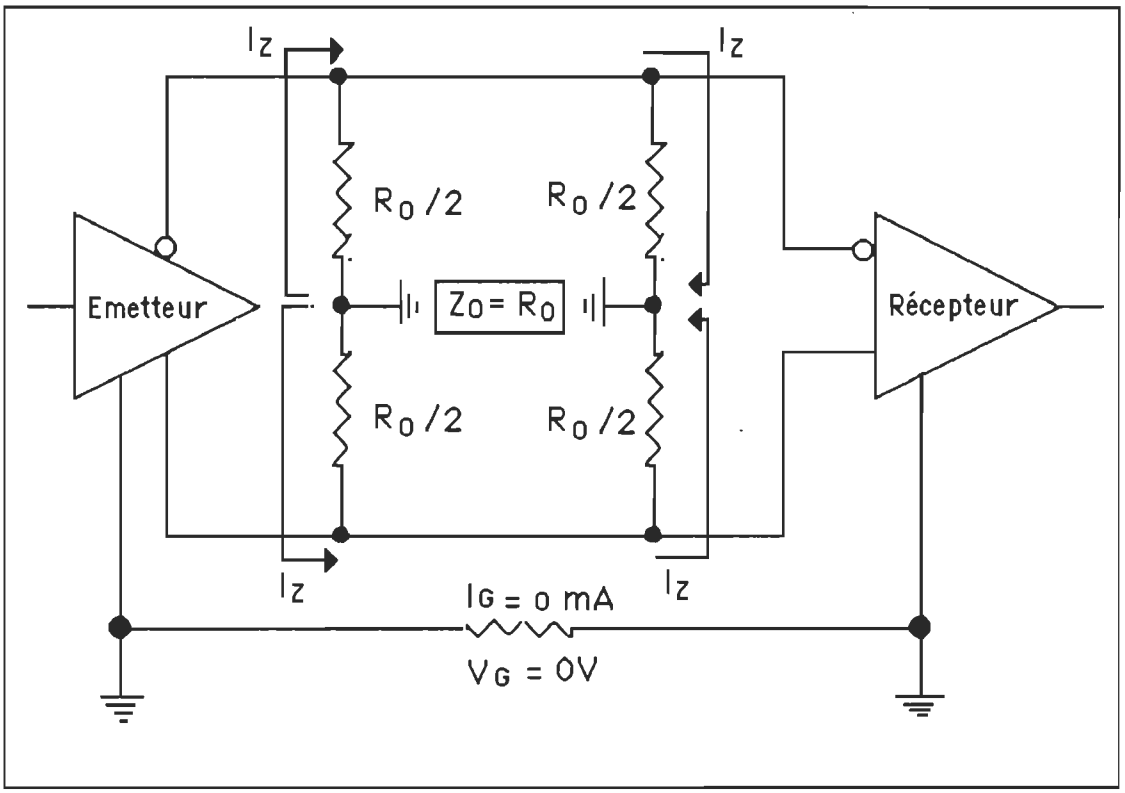


Fig 3.8 . **Ligne équilibrée**

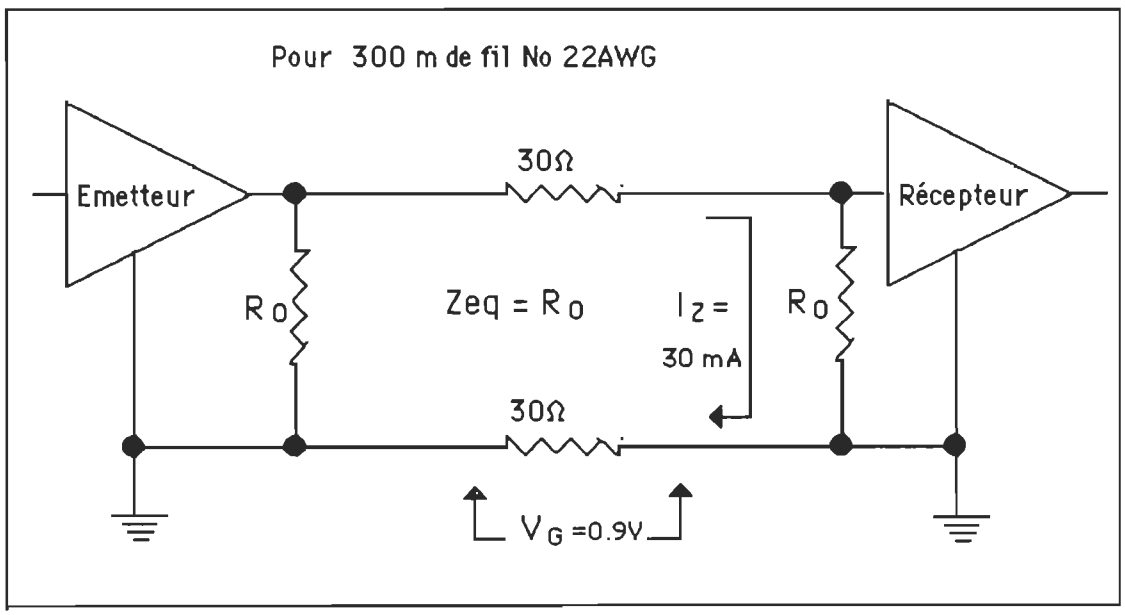


Fig 3.9. **Ligne déséquilibrée**

III-4- Circuit réalisé pour le lien de communication

Le système de communication Fig(3.10) des capteurs intelligents se compose des élements suivants:

- une carte de communication serie ACE- 8250 de National Semiconductor, qui est racordée à la carte mère du micro-ordinateur. Elle effectue des conversions série-parallèle et parallèle-rérie pour la transmission des données, mais il sert surtout pour l'acquisition des données.

- Une entrée/sortie du capteur intelligent SCI.

- Les convertisseurs de niveaux (RS-422) qui transforment les niveaux TTL que l'on trouve dans les systèmes à microprocesseurs ne sont pas compatiples avec les niveaux de tension exigés par les normes des interfaces séries. Ces convertisseurs de niveaux sont présentés par les circuits MC3486 pour les récepteurs et le MC3487 pour les émetteurs de lignes. Ces émetteurs sont commandables pour la haute impédance, pour cette raison les capteurs intelligents raccordés sur le lien peuvent être en permanence en écoute.

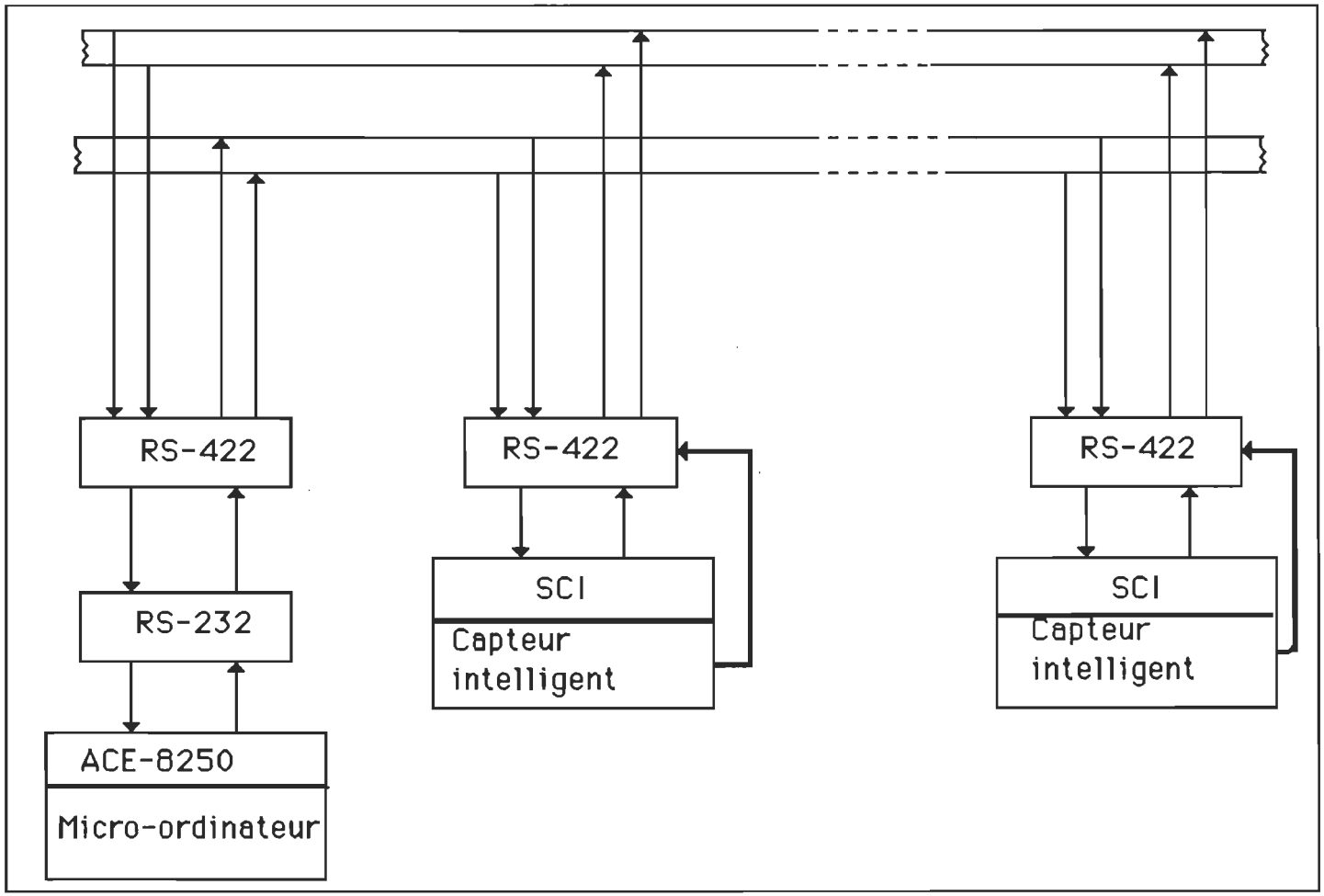


Fig 3.10 Schéma bloc du lien de communication

III-5- Mise en oeuvre de la liaison de données en milieu industriel

Lors d'installation du circuit du lien de communication, pour la réalisaion d'une transmission de données dans un environnement industriel, on doit absolument prendre en compte les phénomènes qui peuvent perturber la qualité de la transmission. Suivant le milieu, les perturbations seront plus ou moins importantes. L'alimentation secteur (ou le probleme de terre) est la cause de perturbations la plus courante pour ce genre d'application.

En effet, les systèmes à base de microprocesseurs ou de composants très sophistiqués, sont très sensibles aux perturbations apportées par l'alimentation secteur, les variations de tension, les micro-coupures, les coupures franches et les parasites. Ces perturbations peuvent entraîner le vieillissement accéléré des composants ou leur destruction, la perte d'informations, des arrêts du système et des erreurs de données.

Lors de la réalisation des liaisons sur des distances importantes supérieures à 200 m, il peut exister une difference de potentiel entre les terres, ce qui entraîne un courant dans la masse, comme cela a été montré à la figure 3.9. Ce phénomène est souvent mal supporté par les systèmes électroniques. Pour résoudre ce problème, il existe plusieurs solutions plus ou moins adaptées. La bonne solution consiste à réaliser un circuit de terre performant avec un piquet de terre ou une ceinture de fond de fouille. L'ensemble des masses des appareils doit être relié en un seul point à la terre du bâtiment. On peut aussi, pour éviter des différences de potentiel entre les

terres, passer un câble de fort diamètre entre la terre du micro-ordinateur et les capteurs intelligents.

CHAPITRE - IV

CONCEPTION DU LOGICIEL DE COMMUNICATION

Les logiciels de communication sont principalement destinés à concourir à la transmission et la réception de l'information. Nous verrons qu'ils constituent aussi un auxiliaire précieux pour le contrôle des données transmises et reçues. Pour engager la conversation de façon adéquate, on a prévu que le logiciel doit contenir au minimum les fonctions suivantes:

- la programmation de l'UART de l'interface série, en précisant la vitesse de trans-mission, le nombre de bits d'arrêt, la nature de la parité choisi et le numéro du port de communication (COM1 ou COM2);

- la programmation du micro-ordinateur pour que chaque sélection d'information, soit transmise à l'interface série, et inversement, que chaque caractère arrivant sur l'interface série soit affiché à l'écran et mémorisé;

- programmation des fonctions de la procédure de commande et de contrôle de la liaison, de protection contre les erreurs et de reprise de l'information utile.

Le programme de communication, qu'il soit simple ou sophistiqué, se compose de deux parties: le module d'initialisation et le module de

communication. Le premier de ces deux modules permet de s'assurer que les éléments du système (le micro-ordinateur, le microcontrôleur et le boîtier ACE-8250) sont compatibles, et prêts à communiquer. Le module de communication a, quant à lui, la lourde charge de mener à bien la communication et d'assurer diverses tâches désirées.

IV-1- Le module d'initialisation

L'initialisation comporte deux séquences: la première permet d'assurer la bonne programmation de l'interface et la seconde, d'assurer que l'interface n'est pas occupé à une autre tâche. Le module doit également avertir l'opérateur de la fin de son exécution.

IV- 1-1- Initialisation de l'UART

La carte d'interface est reliée à des ports d'entrée/sortie: cela signifie que l'ordinateur envoie des caractères et en recoit, à et par l'interface, au niveau d'un port d'adresse donnée. Sur la plupart des micro-ordinateurs cette adresse est fixée, et ne peut en aucun cas être modifiée par logiciel ou plus communément par configuration de straps sur la carte d'interface. Le positionnement de ces straps, force la carte à ne réagir qu'aux signaux effectivement envoyés sur son adresse. Réciproquement, chaque fois qu'un signal en provenance du convertisseur de niveau parvient à la carte, cette dernière le signale au micro en positionnant un drapeau sur son port

d'interface. Dans tous les cas donc, il importe que le logiciel connaisse l'adresse de la carte d'interface, afin d'adresser correctement la communication et d'éviter les erreurs d'acheminement vers d'autres unités du micro-ordinateur.

L'adresse des ports de gestion de l'interface COM1 (port No 1) possèdent les numéros $3F8 à $3FF et ceux de COM2 (port No 2) les numéros $2F8 à $2FF, et dans notre cas l'utilisateur peut utiliser l'un ou l'autre de ces deux ports suivant son propre choix et la disponibilité de ces ports.

Une autre fonction de la routine d'initialisation est la configuration des 7 ports de gestion que chaque interface possèdent (Annexe 2), qui sont en fait les accès aux registres du contrôleur de communication asynchrone. Donc pour programmer l'UART, il nous faut au début le configurer. Ceci fait intervenir le positionnement du diviseur de fréquence dans les registres de diviseurs de fréquence de bit haut et bas, le positionnement du format de communication dans le registre de commande de ligne, le réglage des seuils des sorties de commande de lignes série. On doit charger également le registre de validation d'interruption. Examinons chaqu'un des registres de configuration.

La fréquence de bit est déterminée par les registres diviseurs de fréquence de bit pour adresse $3FB et $3F8. Les deux octets de ces registres sont combinés en nombre de seize bits utilisés pour diviser la fréquence d'horloge fournie au ACE. En conséquence l'ACE exige des fréquences d'horloge inhabituelle, multiples des fréquences de bit asynchrone standard. Le diviseur doit être positionné à seize fois la cadence désirée.

Comme la fréquence de l'horloge pilote doit être divisée par un entier, certaines fréquences de bit (110, 134.5, 1800, 2000, 3600, et 7200) sont légèrement décalées mais l'erreur pour les valeurs données dans la table est toujours inférieurs à 1.5%, ce qui est tout à fait suffisant pour les liaisons asynchrones.

Le format de bits pour la liaison asynchrone est configuré à l'aide du registre de commande de ligne pour adresse $3FB. Les bits de ce registre définissent le nombre de bits par caractère, le nombre de bits d'arrêt et de parité. On trouve également dans ce registre un bit qui peut forcer le 8250 à envoyer le niveau de rupture et un bit de pointeur de registre indirect.

Commençons par les bits 0 et 1, bits de sélection de longueur de mot. Ces bits déterminent combien de bits par caractère l'ACE va émettre et combien il compte en recevoir. Les longueurs de mot possibles sont données dans l'annexe 3. Le bit 2 détermine combien de bits d'arrêt seront envoyés et prévus en réception. Il n'y a qu'un seul bit pour déterminer ce paramètre mais il y a deux possibilités pour chaque longueur de mot et ces possibilités changent; les positionnements de nombre de bits d'arrêt sont donnés à l'annexe.3 . Les bits 3, 4 et 5 commandent le fonctionnement de la parité. Le bit 3 est la validation parité, lorsqu'il est à "1", un bit de parité va être ajouté à tous les caractères émis et la parité va être testée sur tous les caractères reçus. Lorsque le bit 4 est "1", il sélectionné un parité paire. La parité impaire est obtenue avec le bit 4 à "0". Le bit 5 du registre de commande de ligne modifie la modification du bit 4. Lorsque le bit 5, le blocage parité, est à un, un bit de parité sera émis mais ce sera toujours le même que le bit 4. L'ACE testera également la parité reçue

également la parité reçue pour cette valeur. Le blocage du bit de parité à une valeur ou l'autre en permanence supprime l'unité de ce bit pour la détection d'erreur. La mise à 1 du bit 6, force à niveau bas la ligne SOUT et crée donc un état d'arrêt sur la ligne. Cette rupture doit durer au moins 220 ms pour que la majorité des unités centrales reconnaissent la fonction. Le bit 7 positionne l'accès aux mémoires de diviseur de fréquence.

Les trois registres d'état sont utilisés pour déterminer l'état de l'ACE pendant les opérations d'E/S. Ce sont les registres d'état de ligne, d'état de modem et d'identification d'interruptions. Les bits du registre d'état de ligne ont la fonction suivante. Les bits 1 à 4 peuvent être utilisés pour provoquer des interruptions, ou le processeur peut les tester après la réception de chaque caractère, sans utiliser les interruptions. La lecture du registre d'état de ligne remet à zéro les bits 1 à 4. L'instauration du bit 0 indique qu'un caractère a été reçu et doit être lu. Le RAZ de ce bit s'obtient soit par la lecture du registre tampon de réception soit par l'écriture d'un zéro dans le bit 0 du registre d'état de ligne. Les bits 5 et 6 indiquent l'état de l'émetteur. Si le bit 5 est à zéro, un caractère peut être en cours d'émission, mais il y a place dans le registre de stockage d'émission pour un caractère à émettre. Le bit 6 est à zéro lorsque l'émetteur est au repos. Ceci indique que l'émetteur a fini d'envoyer tous les caractères transmis à l'ACE.

Le registre d'identification d'interruptions permet au microprocesseur de déterminer rapidement la cause d'une interruption du 8250. Le bit 0 est utilisé pour indiquer si le 8250 demande une interruption. Si le bit 0 est à un, il n'y a pas d'interruption en attente et les bits 1 et 2 seront à

zéro. Ceci est utile si le 8250 est relié à la même ligne d'interruption que d'autre dispositifs du système. Les bits 1 et 2 représentent l'identité codée de la condition d'interruption la plus prioritaire de l'ACE.

Les deux derniers registres de l'ACE sont le registre de stockage d'émission et le registre tampon de réception. Les données à émettre sont écrites dans le registre de stockage d'émission. Les caractères reçus sont lus dans le registre tampon de réception. Le microprocesseur peut utiliser les indicateurs du registre d'état de ligne ou le système d'interruptions de l'ACE pour prévenir le logiciel système du moment où ces opérations sont nécessaires ou permises. Le module d'initialisation pour la station de contrôle est présente à la fig(4.1) et le programme au complet de ce module est presenté à l'Annexe 2.

IV-1-2- Initialisation de l'SCI

Contrairement à l'initialisation de l'ACE-8250, l'initialisation de l'SCI du microcontrôleur M68HC11 est plus facile puisqu'il contient moins de registres à gérer. Quand le récepteur et/ou le transmetteur est valide, la logique du SCI prend contrôle du registre de contrôle pour les associer au port D. La direction des données des broches RxD et TxD sont respectivement pour l'entrée et la sortie. Même s'il n'a pas le contrôle de la direction des broches du port D, le SCI a le contrôle du registre DDRD, ce dernier peut être important pour une utilisation car il influence le port par une lecture logicielle. L'organigramme de la figure 4.2 montre les étapes d'initialisation du SCI.

Quand le récepteur du SCI est validé (par le bit RE du registre de

contrôle2 SCCR2) le bit 0 de DDRD est surpassé et le buffer de sortie est non validé. Écrire au bit 0 du port D pendant que le SCI a le contrôle de la broche ne change pas l'état logique de la broche, cependant des valeurs écrites sont mémorisées dans un verrou interne. Si le récepteur renonce au contrôle de la broche plus tard, la valeur logique dans ce verrou activera la broche PD0/RxD. Bien que le bit DDRD0 n'affecte pas la broche pendant que le récepteur SCI est valide, il affecte toujours ce qui est retourné, quand le port D est lu. Si DDRD0 est à zéro la broche est lue, s'il est à un la valeur dans le verrou interne du bit 0 du port D est retournée.

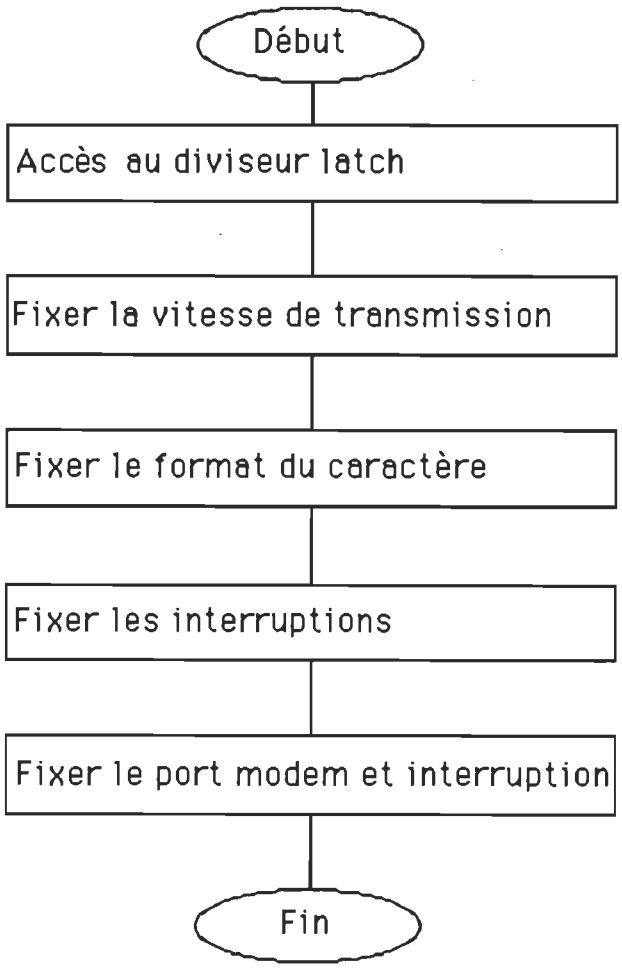


Fig 4.1. Module d'initialisation pour la station de contrôle

Quand l'émetteur est actif, le bit 1 de DDRD est surpassé et le tampon de sortie correspondant devient fonctionnel à cause de la logique SCI (contrairement à la logique du port de sortie). L'émetteur est actif (en contrôlant la broche PD1/TxD) chaquefois que le bit TE (du registre de contrôle2 SCCR2) est à un ou un caractère n'était pas encore transmis quand le bit TE est inactif. Écrire au bit 1 du port D pendant que le SCI a le contrôle de la broche 1, mais il ne change pas l'état logique de cette broche, cependant des valeurs écrites sont mémorisées dans un verrou interne. Si l'émetteur renonce au contrôle de la broche plus tard, la valeur logique dans ce verrou activera la broche PD1/TxD. Si DDRD1 est à zéro la broche est lue, s'il est à un la valeur dans le verrou interne du bit 0 du port D est retournée.

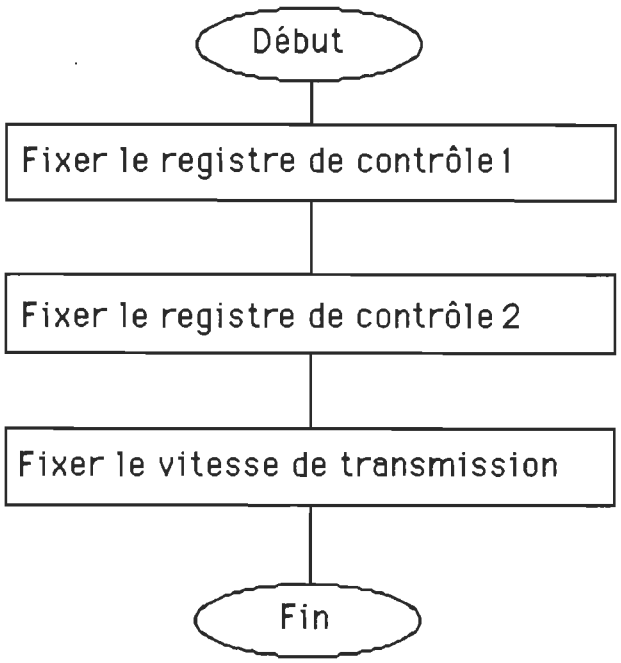


Fig 4.2. Module d'initialisation pour le capteur intelligent

IV-2- Le module de communication

Contrairement au module d'initialisation qui est complexe, le module de communication est de conception facile grâce au module d'initialisation.

IV-2-1- Structure du logiciel de la station de contrôle

Le développement d'une structure implantée au niveau du contrôleur (micro-ordinateur), permettant de réaliser une acquisition de données est un des objectifs primordiaux de cette étude.

Le rôle des routines d'acquisition de données, voir fig 4.3, est de détecter les caractères spéciaux pour les commandes qui nécessitent le traitement de routines particulières et d'afficher les autres. Ces routines empêchent également les événements indésirables, comme la transmission par l'interface d'un caractère de contrôle provoquant la déconnexion de la station en question. La routine de saisie de caractère à partir du clavier doit également vérifier les caractères de contrôle qui permettraient le report.

L'exemple suivant est utilisé dans le but de fournir une explication spécifique de ce que l'on entend par une structure reprogrammable. En supposant qu'un éventuel utilisateur de ce système désire réaliser le contrôle de styrène de chacun des capteurs intelligents, placés à des endroits différents, en fonction des différents paramètres, la température, la pression relative et le temps. En supposant qu'il désire lire chacun de ces capteurs à toutes les 10 minutes, il faudra fournir au système les informations suivantes:

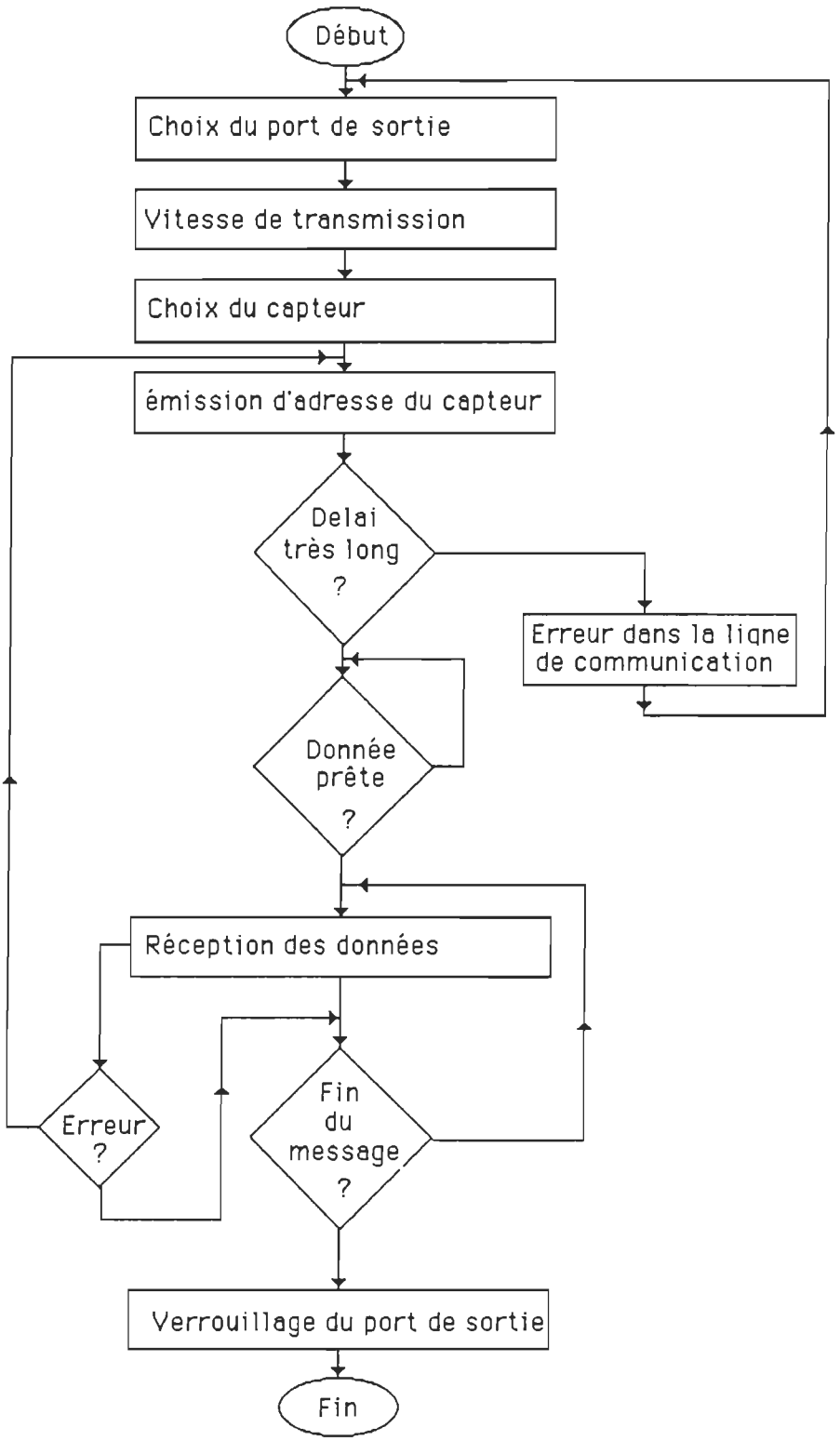


Fig 4.3. Structure minimale du module d'acquisition de données

1. le numéro de l'expérience;

2. le nombre de capteurs sur la ligne;

3. l'heure de la fin d'expérience (après avoir connu l'heure du commencement);

4. la période d'échantillonnage en minutes.

La structure minimale du module d'acquisition de données est montrée à la figure 4.3, cette structure montre les principales étapes à réaliser pour assurer une grande fiabilité d'acquisition de données et une flexibilité au niveau d'une modification du format des messages s'il y a lieu. Les étapes de cette structure sont comme suit:

1. la procédure du choix du port de communication (SETRS232on(1 ou 2)) a été utilisée, cela montre qu'on a le choix entre le port de communication COM1 ou COM2 selon la disponibilité de ces derniers. Et cette procédure assure en plus la bonne gestion de la réception et de la transmision en mode série; pour ce faire il faut initialiser le processus d'interruption, commençant par masquer l'interruption $21 puis on assure la vidange du port d'entrée et on initialise le reste des registres de contrôle de ligne et modem;

2. la procédure pour fixer la vitesse de transmission SET_BAUDE( ), sert à choisir une vitesse de transmission entre le micro-ordinateur et différents capteurs intelligents. Cette vitesse devra être la même pour les deux intervenants de la ligne de communication, sinon on aura une erreur indiquée par le programme principal;

3. la procédure de transmission d'un caractère ENVOI('C') vers le

récepteur d'un capteur intelligent sert à envoyer l'adresse du capteur qu'on veut consulter;

4. une fois l'adresse du capteur en question envoyée, un compteur commence à décompter un délai; si le compteur arrive à la fin du comptage et qu'aucune réponse de la part du capteur n'est arrivée au micro-ordinateur, un message va être affiché sur l'écran qui indique qu'il y a une erreur au niveau du branchement et que la transmission des messages est bloquée. Dans ce cas, on n'a juste qu'à vérifier les broches 2 et 3 du micro-ordinateur avec les broches 20 et 21 du M68HC11. Sinon, on aura des données à recevoir du capteur; l'arriver de ces données va être détectée par la fonction DONNEE_PRETE;

5. si la fonction DONNEE_PRETE est valide, alors la procédure de réception PREND_CARACTERE peut être utilisée pour recevoir des données transmises par le capteur intelligent en question. La procédure PREND_CARACTERE prend caractère par caractère dans la queue de réception par interruption. À la détection du mot de fin des messages, le module d'acquisition prend fin. Après, on purge le registre de réception par la procédure PURGE_QUEUE pour préparer la réception sans encombrement des caractères du prochain message à recevoir;

6. le message reçu par le micro-ordinateur contient l'information sur les valeurs de styrène (en ppm), de température (en °C) et de pression relalive (en kpa). Ces valeurs sont des réels ou des entiers sous forme de 5 caractères. Les erreurs de transmission sont vérifiées par la fonction VAL(ST,X,I) du TURBO-PASCAL version 6.0. Les valeurs reçues sont des chaînes de caractères ST, ces chaînes sont transformées en une valeur de type réel ou entier, en fonction du

type déclaré de X. ST correspond à une valeur numérique. Lorsqu'une erreur se produit, la position dans la variable I est sauvée. Si la transformation se fait sans erreur, I recoit la valeur 0 et X la valeur transformée. Si la transformation se fait avec erreur, on recommence le processus d'acquisition de nouveau, jusqu'à la réception des bonnes données.

Alors, le programme de communication présenté à l'Annexe 2, est organisé de façon modulaire. Les paramètres de base étant déjà présents dans le module d'acquisition de données, ce qu'il nous faut ajouter c'est la possibilité de les modifier à partir du clavier. Il serait ensuite très utile de réaliser quelques-unes des fonctions informatiques propre au micro-ordinateur, comme par exemple le transfert de données ou encore la réalisation de traitement de données reçues. On a ajouté à notre module de communication un grand nombre de modules supplémentaires, sans oublier la possibilité d'ajonctions futures lorsque le besoin s'en fera sentir. On s'est également assuré que le contrôle de l'utilisation de ces nouveaux modules soit bien donné à l'utilisateur de façon aisée (autrement dit, l'interface opérateur machine doit être "sympathique" !).

La façon la plus classique d'établir une interface conviviale avec l'opérateur consiste en la présentation d'un menu recensant les options disponibles à un point de branchement du programme (un point de branchement, comme son nom l'indique, est un point dans le programme où le jeu d'instructions à venir est dépendant de l'option choisie entre plusieurs). Cette méthode de choix par présentation d'un menu est certainement la plus pratique pour l'utilisateur peu familié avec le problème. Pour ce qui nous

concerne, nous choisirons de mener à bien le développement de modules construits autour du concept de sélection à partir d'un menu général. Ce menu est présenté à l'opérateur sous forme d'un menu déroulant, qui contient des rubriques d'un menu principal qui correspondent à des fenêtres, ainsi que les rubriques que ces fenêtres contiennent. Pour obtenir l'information sur les capteurs il suffit que l'opérateur choisisse la bonne rubrique parmis les fenêtres suivantes:

- Acquisition; elle sert à faire l'acquisition de données, les afficher instantanément sous forme de valeurs et de courbes sur le même écran et les mémoriser sous forme de fichiers qu'on peut consulter à tout moment et cela va durer toute la période fixée au début de déclenchement du système;

- Résultats, c'est la consultation des différents fichiers qui sont en mémoire en mentionnant le numéro de l'expérience et le nombre de capteurs. On a le choix entre deux formes de présentation des résultats soit en trois courbes sur le même écran ou bien on présente une courbe à la fois à l'écran.

IV-2-2- Structure du logiciel du capteur intelligent

Pour engager une bonne communication, il faut que le programme implanté au niveau du capteur intelligent soit compatible avec celui de la station centrale. Alors, le protocole choisi doit être bien converti en langage informatique. Parmis les points les plus importants du protocole c'est le format des messages qu'on doit transmettre. L'implantation du protocole choisi est

expliqué par l'organigramme présenté à la figure 4.4.

Le programme en Annexe 3 commence par l'initialisation des registres de l'interface de communication série asynchrone. Ensuite les routines de traitements et de calculs conjointement réalisées [46]. Ces routines ont comme fonctions l'acquisition des signaux analogiques, la conversion de ces signaux en signaux numériques et ensuite le calcul des valeurs de concentration de styrènes, de température et de pression relative.

Lorsque les valeurs calculées sont mémorisées à des adresses respectives, on teste si le capteur choisi peut entrer en communication avec la station de contrôle. Ce test est vérifié par la consultation du registre de réception du M68HC11; si l'adresse reçue correspond bien au capteur en question, on passe à la transmission des valeurs en mémoire. Sinon on recommence la même procédure depuis le début jusqu'à la réception de la bonne adresse.

La transmission des messages vers le micro-ordinateur, se fait de la façon suivante:

- on charge la valeur du styrène qui existe dans la mémoire, sous la forme de 14 bytes, dans le registre de transmission. On remplace les espaces, que la routine de calcul en point flottant fait apparaître au début de chaque valeur pour le signe plus ou moins, par les zéros;

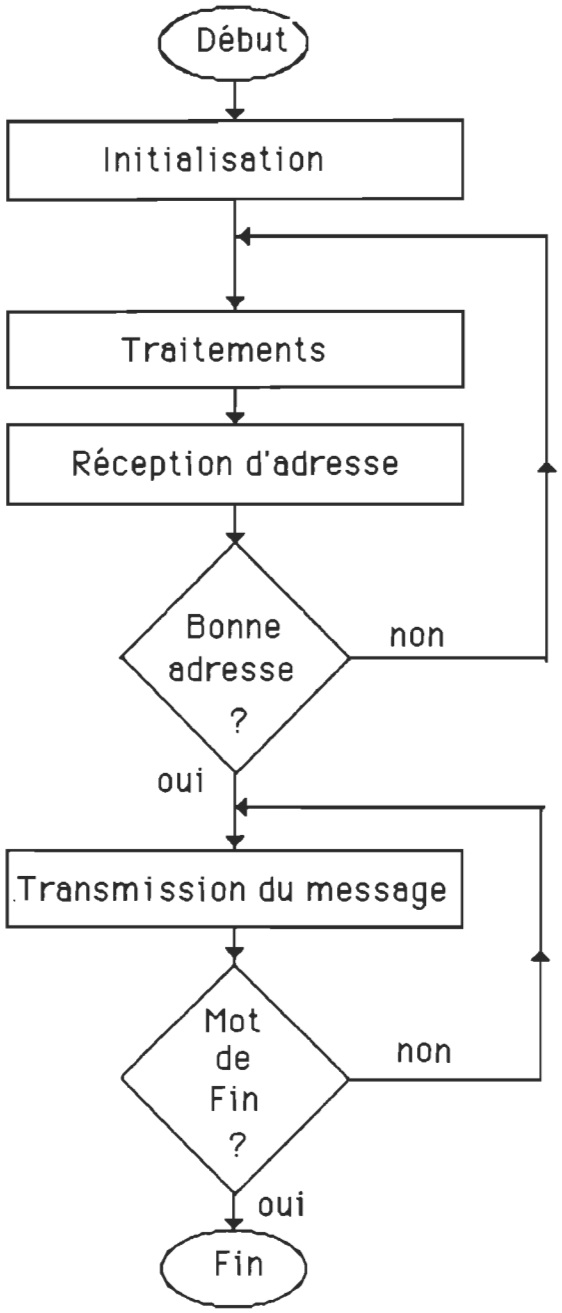


Fig 4.4. Structure du programme au niveau du capteur intelligent

- avant la transmission, on vérifie le port de sortie et on transmet bite par bite. Ces bytes seront séparés par un délai, pour laisser au récepteur du micro-ordinateur le temps de lire les caractères séparément. Cette transmission se poursuit jusqu'à ce que le compteur décompte cinq bytes, cela montre qu'on a transmis juste les premiers cinq bytes par valeur (par exemple 00100 ppm de styrène);

- on va avoir la même procédure pour les deux autres valeurs de la température et de la pression relative. Une fois que ces trois valeurs sont envoyées on envoie un caractère qui va servir de fin du message à recevoir par le micro-ordinateur.

Cependant, cette structure sera changée lorsque plusieurs capteurs intelligents seront connectés au lien de communication. Dans ce cas (fig 4.5), il faut mettre ces capteurs en écoute; pour ce faire on fixe l'émetteur de ligne de ces derniers en haute impédance. La haute impédance est réalisée par la mise à zéro de la broche ENABLE de l'émetteur de ligne, et cette mise à zéro sera effectuée par un des ports de sortie du M68HC11. Alors, lorsque la station de contrôle envoie l'adresse du capteur à qui communiquer, tous les capteurs sur la ligne vont prendre cette adresse mais seul un capteur sera identifié. Le capteur en question enlèvera la haute impédance pour qu'il soit capable de répondre à la station de contrôle par son émetteur. Mais une fois que le message est envoyé la haute impédance est maintenue. Par contre, les autres capteurs non sélectionnés vont maintenir la haute impédance.

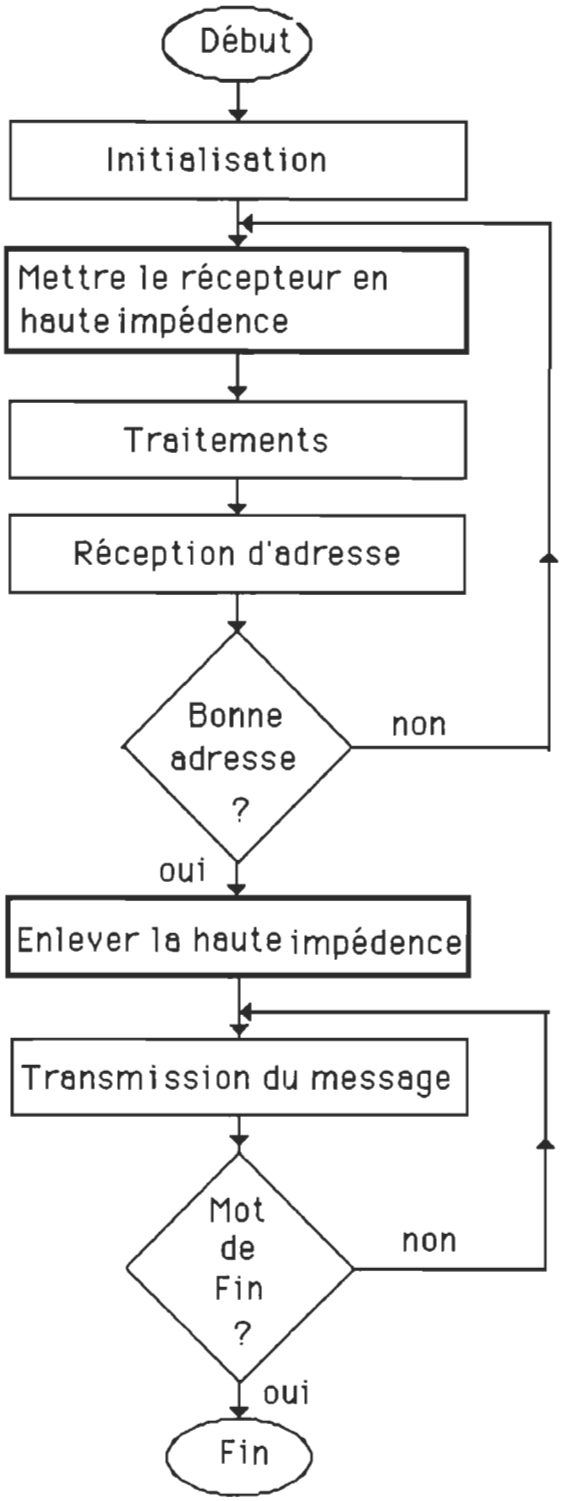


Fig 4.5. Structure du logiciel pour plusieurs capteurs

IV-3- EVALUATION DU LOGICIEL DE COMMUNICATION

Une étude expérimentale a été réalisée dans le laboratoire de génie chimique de l'UQTR, pour faire une évaluation du logiciel de communication des capteurs intelligents du styrène. Cette étude expérimentale consiste à étudier le comportement des vapeurs de styrène dans le temps en fonction des paramètres environnementaux (température et pression). La figure(4.6) montre le système utilisé pour cette étude, il contient un bain thermostaté renfermant une chambre simulant le milieu industriel où on a placé des détecteurs de styrène, de température et de pression, un microcontrôleur pour l'intelligence des capteurs et enfin la station de contrôle (micro-ordinateur).

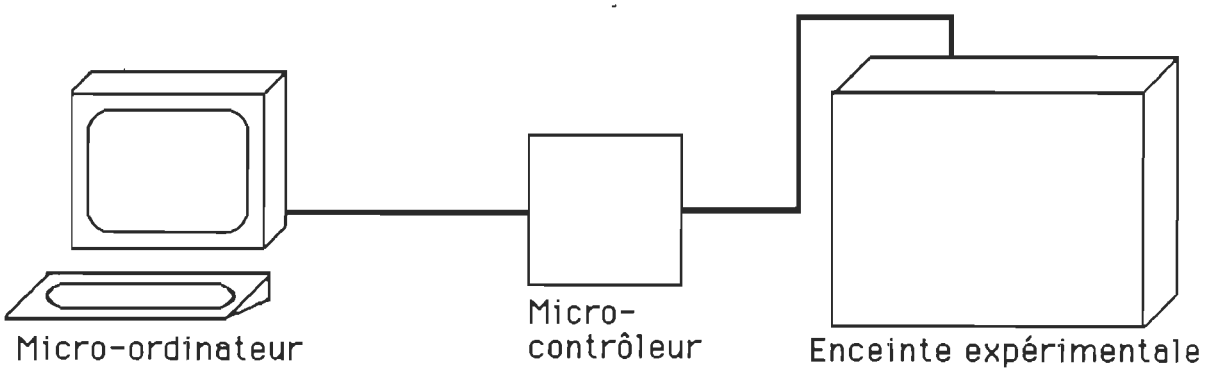


Fig 4.6 Schéma bloc du montage expérimental

IV-3-1- Étude expérimentale

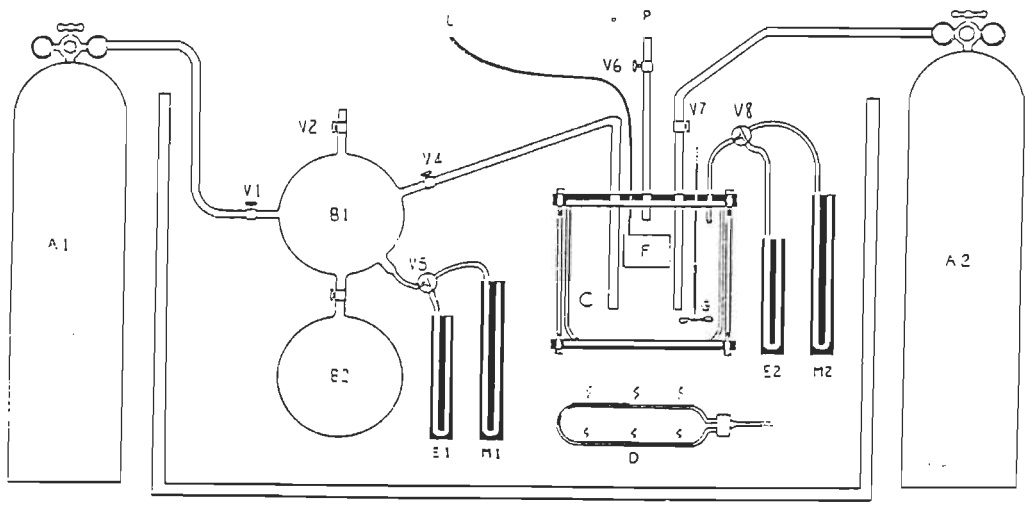


Fig 4.7. Montage du bain thermostaté

Avec le montage du bain thermostaté représenté par la figure 4.7, on est capable d'étudier le comportement du détecteur des vapeurs de styrène dans le temps en variant les paramètres environnementaux susceptibles d'affecter la caractéristique de sortie du capteur en question. Ce montage contient les éléments suivants:

-**C** est la chambre qui simule le milieu industriel et qui est parfaitement étanche;

-**F** représente les détecteurs de styrène, de température et de pression;

-**L** est un câble blindé qui transporte les signaux analogiques du détecteur au micro-contrôleur;

-**G** est l'agitateur qui sert à agiter les vapeurs de styrène à l'intérieur de la cham-bre;

-**V** sont les soupapes à deux voies et à trois voies, qui servent au contrôle des vapeurs dans la tuyauterie. Surtout la soupape V4 qui est micrométrique ce qui nous donne plus de contrôle sur la précision de la valeur de styrène à injecter dans la chambre;

-**B:** le ballon B2 contient du styrène liquide; après la vaporisation ce dernier va s'installer dans le ballon B1;

-**D** sont des résistances pour chauffer le bain à la température désirée, ces résistances sont contrôlables;

-**E** sont des manomètres pour la lecture des concentrations dans la chambre et dans le ballon B1;

-**P** est la purge qui va nous permettre d'évacuer les vapeurs de styrène dans la chambre C.

Le principe de cette étude expérimentale, est d'injecter différentes concentrations des vapeurs de styrène dans la chambre C, et de voir le comportement du détecteur de styrène par rapport aux conditions climatiques d'un milieu industriel. Pour injecter différentes concentrations des vapeurs de

styrène dans la chambre C, on procèdé par deux méthodes: la première méthode, est le jeu des manomètres et la deuxième est l'injection par seringue.

IV-3-1-1- Manipulation par le jeu des manomètres

On place du styrène liquide dans le ballon B2 et on ferme les soupapes V1, V2 et V4, une fois que tous les éléments du bain sont immergés dans l'eau on chauffe le bain avec les résistances chauffantes à une température voulue. La fixation d'une température du bain provoquera lévaporation du styrène liquide. On maintient le ballon B1 et la chambre C à une pression constante; après on ouvre la soupape V3 qui sépare les ballons B1 et B2, et les vapeurs de styrène prendront place dans le ballon B1. Normalement il y aurait une faible augmentation de la pression qui n'est pas détectable par le manomètre de mercure M1; pour permettre la lecture de cette variation de pression on commute le manomètre de solvant E1 avec la soupape V5.

En ouvrant la soupape micrométrique V4, on injecte une concentration aléatoire de styrène dans la chambre C puis on ferme cette soupape. Ensuite, la lecture de la variation de pression qu'on avait notée dans le manomètre E1 va nous permettre de calculer la concentration du styrène qu'on avait injecté dans la chambre C. On recommence l'injection deux à trois fois pour chaque expérience. Après chaque expérience, on fait une purge totale du système. Cette purge nous permet d'évacuer toutes les vapeurs de styrène qui se trouvent dans le ballon B1, dans la tuyauterie et dans la chambre C. Pour cela on ouvre les soupapes V2, V4 et V6; on doit s'assurer aussi que les

soupapes V5 et V8 sont commutées sur les manomètres de mercure. Ensuite, on fait circuler de l'air dans tout le système, en utilisant les deux bouteilles d'air A1 et A2, et en ouvrant les soupapes V1 et V7.

IV-3-1-2- Manipulation par injection directe

Pour l'injection directe ou par seringue, il faut apporter des modifications au montage de bain thermestaté montré à la figure 5.2. Parmi les modifications effectuées, on débranche la liaison qui relie le ballon B1 et la chambre C. A la place de cette liaison, on a placé un Septum-Nut qui servira à la seringue et une membrane septa qui permettera de garder l'étanchéité de la chambre C après injection. Pour injecter différentes concentrations dans la chambre C, on utilise une seringue de l'ordre du microlitre et il faut:

- premièrement, fermer toutes les soupapes ouvertes; couper la circulation de l'air dans la tuyauterie et maintenir une pression constante dans la chambre C. Avant l'injection il faut attendre que la température et la concentration de styrène dans la chambre C soit stable pendant un certain temps donné.

- deuxièment, purger après chaque expérience pour éliminer les vapeurs de styr-ène qui se trouvent dans la chambre C et dans la tuyauterie. Pour cela, on ouvre la soupape V6 et on doit s'assurer que la soupape V8 est commutée sur le manomètre de mercure M2. Pour faire circuler de l'air dans la chambre C, ensuite, on fait circuler de l'air dans tout le système, en utilisant la bouteille d'air A2, et en ouvrant la soupape V7.

Pour les deux méthodes d'injection et pour chaque expérience, il faut mettre en marche la station de contrôle (micro-ordinateur). Pour ce faire on exécute le programme principal et il va apparaître sur l'ecran un menu déroulant dans lequel on trouve cinq fenêtres. En choisissant la fenêtre traitement, dans laquelle il faut donner quelques informations au système; dont:

- le jour et la date de l'expérience sont données par le micro-ordinateur;

- il faut entrer le numéro de l'expérience;

- entrer aussi le nombre de capteurs connectés sur le lien de communication;

- l'heure du commencement de l'expérience sera donné par le systéme;

- il faut donner l'heure d'arrêt du système, sans jamais dépasser 24 heures;

- enfin on choisit une période d'échantillonnage pour la prise des données en minutes.

Après que le micro-ordinateur ait reçu les commandes correctement et après chaque periode de prise de données qu'on avait choisie, les valeurs de la concentration du styrène (en ppm), de la température (en °C) et de la pression relative (en kpa) prises dans la chambre C en fonction de temps, seront affichées à l'écran sous forme de valeurs instantanées et des courbes en fonction du temps. À chaque période d'échantillonnage, l'évolution des ces trois valeurs sera presentée à l'écran et aussi la valeur moyenne du styrène sera affichée. Avec cette valeur moyenne on peut intervenir chaque fois qu'elle dépasse les 50 ppm pendant une periode de 8 heures, cette intervention sera sous forme des bips d'ordinateur.

IV-4- Résultats expérimentaux

Les résultats expérimentaux sont présentés à l'Annexe 7, par les numéros d'expériences, le jour et la date lorsqu'on a effectué ces expériences. Les expériences 1 à 9 et 17 à 19 ont été efféctueés par la méthode d'injection par seringue par contre, celles des numéros 10 à 16 ont été efféctués par la méthode de jeu des manomètres.

Ces résultats montrent parfaitement le comportement des vapeurs de styrène en fonction des paramètres environnementaux (temperature et pression) et du temps. Pour les expériences 1 à 9 (voire Annexe 7) ou à la figure 4.8, on a injecté directement dans la chambre C du styrène liquide, et on a opéré à une température allant de 25°C à 65°C. On remarque que le détecteur de styrène réagit continuellement dans temps avec les vapeurs de styrène, car le styrène liquide va s'évaporer avec le temps en fonction de l'augmentation de la température et à une pression relative de la chambre C oscillant entre 0 et 1.0 kpa. En injectant une forte concentration de styrène liquide au début de chaque expérience, on ne peut pas voir une nette variation de concentration si on en injecte par la suite car, on a assez de styrène liquide dans la chambre C, et il lui faut aussi beaucoup de temps pour s'évaporer. C'est pour cette raison qu'il faut injecter des petites concentrations de styrène liquide et choisir une grande période d'échantillonnage de l'ordre de 15 à 30 minutes. On a noté des variations de concentrations à 12H00 de l'expérience N°1, à 2H30 et à 3H45 de l'expérience N° 3, à 14H30 de l'expérience N° 7 et à 17H00 de l'expérience N° 19.

Pour les expériences 10 à 16 (voire Annexe 7) ou à la figure 4.9, on a utilisé la méthode de jeu des manométres. Avec cette méthode et grâce à la soupape micrométrique V4, on peut injecter à un instant donnée une

concentration de vapeur de styrène dans la chambre C. D'après les résultats des l'expériences 14 et 15, on remarque une nette variation de concentration de styrène dans la chambre C au moment même de l'injection. Cela est dû à une stabilité de la concentation de styrène dans la chambre C, ou bien à une répartition uniforme des vapeurs dans toute la chambre C. Et lorsqu'on injecte de la vapeur de styrène, la variation de concentration dans la chambre C est vite détectée par le capteur de styrène.

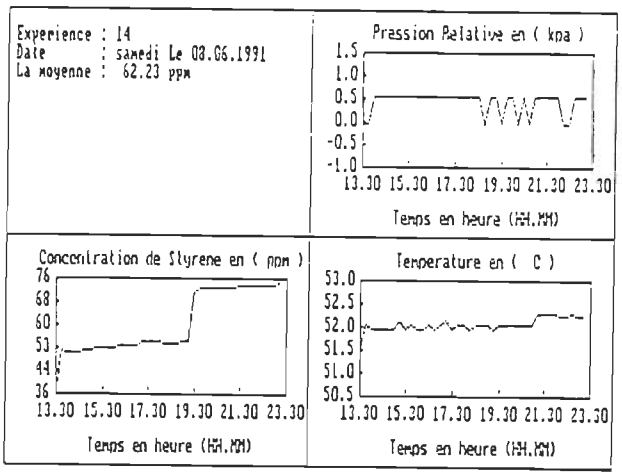


Fig 4.8 Injection directe

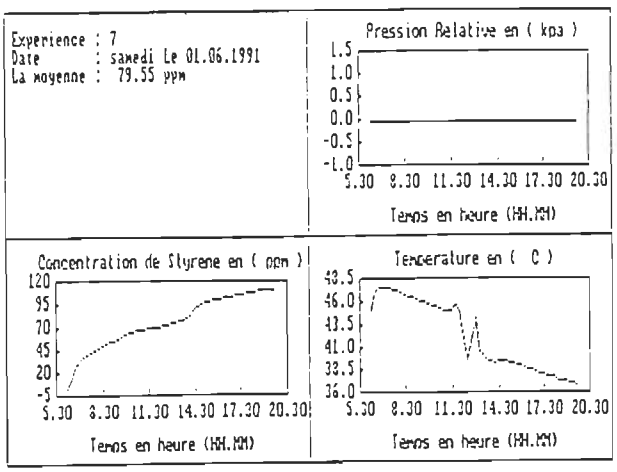


Fig 4.9 Jeu de manomètre

CHAPITRE V

CONCLUSION

Dans le but d'effectuer une évaluation du projet, une revue des différents objectifs mentionnés dans l'introduction est réalisée. On constate, entre autres, que:

1- Une stratégie de communication bidirectionnelle a été developpée, pour permetre l'échange entre un micro-ordinateur central de contrôle et un réseau de capteurs intelligents (microcontrôleurs M68HC11).

2- La procédure d'acquisition des données est entièrement accomplie par le micro-ordinateur, ce dernier n'a qu'à recueillir les différents résultats et de les mémoriser.

3- La procédure d'acquisition est reprogrammable et peut être regonfigurée par une simple modification des modules de communication.

4- L'installation des différents points sur le réseau est simple, car le support physique de communication choisi est une paire de 2 fils. Chaque capteur intelligent est alors ajouté en parallèle sur le lien.

5- La structure topologique "BUS" ainsi que le protocole utilisé assurent une très bonne fiabilité lors de tout échange d'information sur le lien.

6- La structure d'échange entre le micro et le lien est réalisée par le logiciel de communication conçu pour cette application, qui est d'une stucture simple pour permettre à l'usager d'accéder à toute information sur les capteurs intelligents.

7- Finalement, le seul point qui reste à réaliser est celui de raccorder d'autres capteurs intelligents en parallèle avec le lien de communication; le test a été fait sur deux cartes EVB-M68HC11 et une carte EVM-M68HC11, sur lesquelles on a simulé des vecteurs de 16 caractères chacun. Puis on a raccordé sur le même résau ces trois systèmes au micro-ordinateur, la communication s'est déroulée sans aucun problème. Cela signifie que le logiciel gére très bien le lien de communication.

L'analyse de chacun de ces points, illustrant un aspect des différents objectifs mentionnés au début de cette étude, laisse apparaître que la presque totalité d'entre eux ont été atteints.

Un des avantages importants du protocole développé réside au niveau du format des messages proposés. En effet, le fornat utilisé possède des caracteristiques de fiabilité, ainsi qu'une facilité de changement si désiré ou des modifications dans le futur.

Par ailleurs, un avantage au niveau de la fiabilité du protocole serait de nature à favoriser son utilisation dans des milieux électriquement hostiles

(où la présence de bruit est importante) ou dans des milieux où l'intégrité de l'information reçue est essentielle.

Un autre avantage très important a trait à la facilité de reconfigurer le système même après l'installation.

En fait, si on analyse la structure du logiciel de communication, on constate que, dû à la variabilité de la longueur des messages, il est aisé de générer un message de n'importe longueur. À ce moment, il suffirait de modifier les caractères de contrôle ou des fins des messages afin de transmettre les données selon le nouveau format de message utilisé. Mais aucune modifications au niveau de la réception ne serait nécessaire.

Un inconvénient mineur du système réside au niveau du taux de transfert qui ne peut excéder 9600 bauds/s avec l'utilisation d'une UART et d'un SCI. Mais pour cette application cela n'affecte aucun résultat.

Une autre limitation vient de fait que la mémoire vive du micro-ordinateur doit être assez élevée (l'ordre de grandeur de cette memoire dependra de nombre du capteurs à raccorder sur la ligne) pour mémoriser 4 fichiers (styrène, température, pression et le temps) et cela pour chaque capteur connecté sur la ligne.

BIBLIOGRAPHIE.

[1] Atlas, E. and Giam,C.S., Science (USA), Vol. 211, No 4478, pp 163-165, 1981.

[2] Ackerman, M., Frimont, D., Muller, C., and Webbles, O.J., Pure and apple. Geophys. (Switzerland), Vol. 117, No. 3, pp 367-380, 1978.

[3] Thurston, G.D., and Lioy, P.J, Atoms. Environ. (GB), Vol. 18, No. 3, pp 687-698, 1987.

[4] Endlish, R.M et al., Atoms. Environ. (GB), Vol. 18, No. 11, pp 2345-2380, 1984.

[5] Santodomnato, T. et al., Monograph on Human Exposure to Chemicals in the Workplace : Styrene. Center for chemical Hazard Assesment. Syracuse Research Corporation. Report No SRC-TR-84-1124, July 1985.

[6] Coulter, K. E. Kehde,H. and Kiscokk, B.F., Styrene and related Monomers "in" Vinyl and dienes Monomers, part. 2, Wiley, pp 479-476,1971.

[7] Brown, W.D., Arzona Medecine, Vol. 41, No. 7,pp 474-478, 1984.

[8] Meretoja, T., Vainio, H. and Jarventans, H., Toxicology Letters, Vol. 1,No. 5-6,pp 315-318, 1978.

[9] Anonym, Styrene, Industriel Chemecal Exposure : Guidelines for Biological Monitoring, Biomidical Puplications, Davis, cal. USA, pp 71-80, 1983.

[10] Bedor, M. Doroz, P-O and Guillemin, M., International Archive of Occupational and Environmental Health, Vol.55, No. 4, pp 331-336, May 1985.

[11] Guillemin, M. and Beaur, D. International Archves of Occupationaland Enviromental Health, Vol. 44, No. 4, pp 249-263, 1979.

[12] Lof, A. E. and Al., Scandinavian Journal of Work, Environment and Health, Vol. 12,No. 1, pp 70-74, 1986.

[13] Karbowski, R. J. and Braum, W. H., Journal of Chromatography, Vol. 160, pp 141-145, 1978.

[14] Lorimier, W. Styrene and Butadiene. Environmental and Occupational Medecine, pp 633-637, 1983.

[15] Anonym. Styrene in air. Laboratory Methode Using Porous Polymer Absorbent Tubes, Termal Desorption and gaz chromatography, Health and Safety Execlusive Occupational Medecine and Hygiene Laboratory, pp1-4, 1983.

[16] Van Roosmalen, P.B. and Drummond, I., British Journal of Industriel Medecine, Vol. 35, pp 56-60, 1978.

[17] Ogata, M. Sugihara, R., Japanese Journal of Industriel Health, Vol.20, pp 218-219, 1978.

[18] Van den Heod, N. van Asselen, O. and van Dogen, Am. Ind. Hyg. Assoc. J. 48(3). pp 252-256 , 1987.

[19] Loper, G. L., Sasaki, G.R., and Stamps, M.A., Appl. Opt., Vol. 21, No. 9, pp 1648-1653, 1982

[20] Todd, W.F. and Shulman, S. A., American Hygiene Association Journal, Vol. 45,No. 12, pp 817-825, 1984.

[21] Darron, B., design News, pp 24-25, 1987.

[22] Giachino,J.M.,Journal of Physics, No.19,pp 891-896, 1986.

[23] Haskard, M.R. "**Microelectronic Sensor Technology**". Journal of physics, Section E, Vol. 19, pp 891-896, 1986.

[24] Favennec J. -M., "**Smart Sensor in Industry**", J,Ph. : Sc. Instrum, Vol. 20, No. 9, 1987.

[25] Hal Halman, "**Intelligent Sensor : The Merging of Electronics and Sensing**", Technical Insight, Inc. Englewood/Fort Lee, NJ.

[26] Wise, K.D., "**The Role of Thin Films in Integrate Solide-State**" pp 617-622, 1986.

[27] Intersil, Remdac Evaluation KIt technical manual, Intersil Inc., USA, 1980.

[28] Persun T. , and coll., "**Data acquisition or a bord: What`s happening** ", Instruments & Control Systems, Vol. 55, No. 3, mars 1982.

[29] Hampson, P. , "**High-speed data acqisition an signal processing**", New Electronics, London, 1981.

[30] Data translation Inc., "**Data acquisition system provides continous throughput**", Electrical Design News, Boston, 12 mai, 1982.

[31] Chester, M., "**Local Intelligence Expand the Functions of data-acquisition systems for industry- Industrial use**", Electronic Design, Vol. 30, No. 9, 1982.

[32] Weitzman, C., **Distributed Micro/Minicomputer Systems**, Prentiice-Hall Inc. Englewood Cliffs, N.J., USA, 1980.

[33] Myer, W., "**Toward a local network standard**", IEEE Micro, Vol. 2, No. 3, 1982.

[34] Goldberger, A. and coll., "**Small-area networks Fit jobs too small for local nets**", Electronics, Vol. 55, No. 22, 1982.

[35] Macci, G. Guilbert, J.F , "**Téléinformatique**", Collection Dunod Informatique, paris

[36] Lilen , H., " **Interfaces pour microprocesseurs et micro-ordinateurs**", Edition Radio, Paris, 1983.

[37] Morvan, C. Carto, M., " **Liaisons et transmission serie** " , Editions, Paris, 1988.

[38] Pujolle, G. Schwartz, M., "**Réseaux locaux Informatiques**", Eyrolles, Paris, 1989.

[39]. Nussbaumer, H. "**Téléinformatique I**", Collection informatique, 1982.

[40]. Lorains, "**Réseaux téléinformatiques**", Polytechnique de lorrain, 1979.

[41]. Sinnema, W. McGovern,T., "**Digital, Analog and Data Communication**", Prentice-Hall, 1986.

[42]. Atkinson, J.K , "**Communication protocols in instrumentation**", J.Phys.E.:Sci.Instrum., No 20,Vol 5,1987.

[43]. Munier, J.M, "**Introduction à la téléinformatique**". Collection pratique de l'informatique, Eyrolles, 1984.

[44]. Sénécal, S. "**Interfaces et communication: Interface série V24-RS232c**", "Toute l'électronique", No 496 août-septembre 1989.

[45]. Goldberger, A., and Lau, S.Y., "Understanding datacom protocols by examining their structures", EDN, 3 mars, 1983.

[46]. Elhiri, Mohammed. , "Étude et développement d'un capteur intelligent pour la détection des vapeurs de styrène", U.Q.T.R.

ANNEXE 1

```
program  simulation;

uses Crt,
     Printer;

const
     NBSTATION = 4;


var
     prob_emission,
     val_max,
     nombre_moyen_paquets_en_attente,
     nombre_total_paquets_en_attente:     array [1..NBSTATION] of
                                                          real;
     paquets_en_attente,  prochain_cycle_pour_emettre,
     nombre_paquets_generes,  nombre_de_collisions,
     nombre_max_paquets_en_attente: array [1..NBSTATION] of
                                                  integer;
     nouveau_paquet_a_emettre,  paquet_a_emettre:  array
                                    [1..NBSTATION] of boolean;
     duree_simulation,  max_aleat,  cycle_courant,  nostation,
     nombre_total_cycles_avec_collision:  integer;


{*************************************************}

procedure  lire_params;

{   Cette procédure a pour fonction de lire les paramètres de la
simulation.
    Ces paramètres sont la durée de la simulation en nombre de cycles
d'émission,  la valeur de max_aleat, qui représente l'intervalle
d'attente le plus long et la probabilité d'émission associée à chaque
station

}


var
```

```
    nostation, ligcour: integer;

begin {procedure}

    ClrScr; GotoXY(10,5);
    writeln('Durée de la simulation ? : ');
    GotoXY(45,5);
    read(duree_simulation);
    GotoXY(10,6);
    writeln('longueur max de l''attente ?: ');
    GotoXY(45,6);
    read(max_aleat);
    ligcour := 8;
    for nostation := 1 to NBSTATION do
    begin
        GotoXY(10,ligcour);
        write('Probabilité d''émission de la station ',nostation);
        GotoXY(60,ligcour);
        readln(prob_emission[nostation]);
        igcour := ligcour + 1
    end {for}
end; {procedure}


{************************************************}

procedure  calcule_intervalle;
```

{ Cette procédure permet l'implantation des probabilités d'émission. Lors de la simulation, des nombres aléatoires seront générés entre 0 et 9999.

Une plage entre ces deux nombres, de taille correspondant à la probabilité désiré est associée à chaque station. Si un des nombres tirés se situe dans la bonne plage pour une station, cette station a un paquet à émettre.

Si la valeur aléatoire tirée ne correspond à aucune des plages de toutes les stations, alors il n'y a pas de nouveau paquet à émettre pendant ce cycle. On s'appuie ici sur le fait que la génération de nombres aléatoires est uniforme. De cette façon, en procédant par intervalles, on obtient une intégrale entre val_max[i-1] et val_max[i] correspondant à la probabilité désirée.

```
}

var
   nostation: integer;

begin {procedure}
   for nostation := 1 to NBSTATION do
   val_max[nostation] := (prob_emission[nostation] * 65535) - 1.0
   {end for}
end; {procedure}


{*************************************************}

procedure  init_simulation;

{ Cette procédure initialise les valeurs de départ de la simulation. }

var
      nostation: integer;

begin {procedure}
   randomize;                    { on veut générer des séries différen-
                                          tes à chaque
                                          simulation  }
   for nostation := 1 to NBSTATION do
   begin
      nouveau_paquet_a_emettre[nostation]          := FALSE;
      paquets_en_attente[nostation]                := 0;
      prochain_cycle_pour_emettre[nostation]       := 0;
      nombre_max_paquets_en_attente[nostation]     := 0;
      nombre_total_paquets_en_attente[nostation]   := 0;
      nombre_moyen_paquets_en_attente[nostation]   := 0;
      nombre_de_collisions[nostation]              := 0;
      nombre_paquets_generes[nostation]            := 0
   end; {for}
   nombre_total_cycles_avec_collision  :=  0;
end; {procedure}


{*************************************************}
```

```
procedure generer_nouveaux_paquets;

var
      nostation        : integer;
      code_aleatoire   :  word;

   { Cette procedure tire un nombre pseudo-aléatoire pour chaque
station. Selon la valeur tirée, on détermine si la station a un nouveau
paquet à émettre.  On détermine ainsi si chacune des stations a un
paquet à émettre pour l'intervalle de temps courant. On s'appuie ici
sur le fait que la génération de nombres pseudo-aléatoires possède
une distribution uniforme et que si on choisit aléatoirement parmi
des nombres aléatoires, on conserve une distribution uniforme. Pour
cette raison,il n'est pas vraiment nécessaire d'utiliser des
générateurs pseudo-aléatoires différents.

   }

begin {procedure}
   for nostation := 1 to NBSTATION do
   begin
      code_aleatoire := random(65535);
      if code_aleatoire < round(val_max[nostation]) then
      begin
         nouveau_paquet_a_emettre[nostation] := TRUE;
         nombre_paquets_generes[nostation] :=
         nombre_paquets_generes[nostation] + 1
      end
      else
      nouveau_paquet_a_emettre[nostation] := FALSE; {end if}
   end {for}
end; {procedure}



{**************************************************}

procedure  determiner_stations_autorisees;
```

```
    { Cette procédure permet de déterminer quelles stations sont ont
quelque chose à émettre lors du cycle courant. Pour ce faire, on
vérifie qu'on a soit un paquet en attente dont le délai est écoulé ou
encore un nouveau paquet à émettre. Dans le cas où on a un nouveau
paquet à émettre, on vérifie s'il y a des paquets en attente. Si c'est le
cas, on met le nouveau paquet en attente et on considère ne plus avoir
de nouveau paquet à émettre.

    }

var
    nostation: integer;

begin {procedure}
    for nostation := 1 to NBSTATION do
    begin
        { initialisation des vecteurs pertinents }
        paquet_a_emettre[nostation]   := FALSE;
        if paquets_en_attente[nostation] > 0 then
        begin
            { Si paquets déjà en attente }

            if prochain_cycle_pour_emettre[nostation] = cycle_courant
            then begin
                { si l'attente d'un paquet est terminée }
                paquet_a_emettre[nostation]    := TRUE
            end {if}
            else begin
                { si l'attente du paquet n'est pas terminée }
                if nouveau_paquet_a_emettre[nostation] then
                begin
                        { on considère que si le nouveau paquet ne peut
                    être émis à ce cycle parce que des paquets sont déjà
                    en attente, alors on mettra ce nouveau paquet en file
                    et on ne le considérera plus comme un nouveau
                    paquet au niveau de la résolution de collisions

                        }
```

```
                    paquets_en_attente[nostation] :=
                    paquets_en_attente[nostation] + 1;
                    nouveau_paquet_a_emettre[nostation] := FALSE;

                    { mise à jour du compteur de paquets max en attente
                    }
                    if  nombre_max_paquets_en_attente[nostation]
                        < paquets_en_attente[nostation] then
                        nombre_max_paquets_en_attente[nostation] :=
                            paquets_en_attente[nostation];
                    {endif}
                end; {if}
                paquet_a_emettre[nostation]    := FALSE
            end {if}
        end
        else begin
            { Si aucun paquet en attente }
            if  nouveau_paquet_a_emettre[nostation] then
                paquet_a_emettre[nostation] := TRUE
            else
                paquet_a_emettre[nostation] := FALSE
        end {if}
    end {for}
end; {procedure}



{*************************************************}

function  situation_de_collision: boolean;


    { Cette petite fonction fait le tour des stations afin de
déterminer si lors d'un cycle particulier, plus d'une station sont
autorisées à émettre et ont effectivement quelque chose à émettre.
Si c'est le cas, la fonction retourne TRUE

    }

var
    nostation, nombre_stations_emettrices:     integer;
```

```
begin {function}

    { on calcule ici le nombre de stations autorisées ayant quelque
    chose à émettre }

    nombre_stations_emettrices := 0;
    for nostation := 1 to NBSTATION do
        if paquet_a_emettre[nostation] then
            nombre_stations_emettrices :=
            nombre_stations_emettrices + 1;
        {end if}
    {end for}
    { si on a collision, on retourne TRUE et on met à jour le compteur
    de collisions }

    if nombre_stations_emettrices > 1 then
    begin
        situation_de_collision := TRUE;
        nombre_total_cycles_avec_collision
            := nombre_total_cycles_avec_collision + 1
    end
    else
    situation_de_collision := FALSE {end if}
end; {end function}



{*********************************************}

procedure traiter_collision;

var
    nostation: integer;

    { Cette procédure s'occupe du traitement des collisions. On fait le
tour de toutes les stations autorisées ayant un paquet à émettre. Pour
chaque cas, si le paquet était déjà en attente, on lui assigne un
nouveau délai d'attente avant réémission. Si le paquet était un
nouveau paquet, alors on le met en file. On vérifie si le nombre de
paquets en attente dépasse le maximum courant et si c'est le cas, on
```

```
met à jour le maximum.

    }

begin {procedure}
    for nostation := 1 to NBSTATION do
    begin

        { on tire un nombre aléatoire situé entre 1 et max_aleat pour
        chacune des stations touchées par la collision. Si la
        collision implique un nouveau paquet à émettre, on mettra le
        paquet en attente. S'il s'agit d'un paquet déjà en attente, on
        retardera son émission. Qu'il s'agisse d'un nouveau paquet ou
        d'un paquet déjà en file, on devra tirer un nouveau délai
        étant donné la collision }


      if paquet_a_emettre[nostation] then
      begin

          {  S'il s'agit d'un nouveau paquet qui entre en collision
        et non un paquet en réémission, on met ce paquet dans la
        file. Dans le cas où il y avait un nouveau paquet mais qu'il y
        avait déjà des paquets en attente, le paquet a déjà été mis
        en file dans la procedure Déterminer_stations_autorisées.

          }

        nombre_de_collisions[nostation]
            := nombre_de_collisions[nostation] + 1;
        if nouveau_paquet_a_emettre[nostation] then
        begin
            paquets_en_attente[nostation] :=
            paquets_en_attente[nostation] + 1;
            if nombre_max_paquets_en_attente[nostation]
                < paquets_en_attente[nostation] then
                nombre_max_paquets_en_attente[nostation] :=
                    paquets_en_attente[nostation];
            {endif}
        end; {if}
```

```
            {   On tire maintenant un nombre aléatoire qu'on
            additionnera au cycle courant pour déterminer à quel cycle
            la station pourra émettre de nouveau.

            }
            prochain_cycle_pour_emettre[nostation] := cycle_courant +
                                                      random(ma
                                                      x_aleat) +
                                                      1;
            paquet_a_emettre[nostation]  := FALSE;
        end {if}
    end {for}
end; {procedure}



{**********************************************}

procedure  traiter_emission;

{   Cette procedure traite l'émission d'un paquet dans le cas où
aucune
    collision n'a lieu. La détermination de la situation de collision de
    même que son traitement sont effectués dans les modules
    Situation_de_collision et Traiter_collision. Le traitement lui
    même consiste à déterminer quelle station est la station
    émettrice, à déterminer si lepaquet à émettre est un paquet qui
    vient d'être généré ou un paquetqui était en attente et
    décrémenter la file lorsque c'est le cas. A ce moment, on émet un
    délai d'attente pour le prochain paquet dans la files'il en reste.

}


var
    nostation, station: integer;

begin
    station := 0;
    for nostation := 1 to NBSTATION do
        if  paquet_a_emettre[nostation]  then
```

```
        station := nostation; {endif}
    {endfor}
    if station <> 0 then
    begin
            {       paquet à émettre lors du cycle courant mais on a
        déterminé précédemment qu'il n'y avait pas collision et que
        l'émission était autorisée pour cette station. On va donc
        émettre le paquet

            }

        if paquets_en_attente[station] > 0 then {si paquets en attente}
        begin
            paquets_en_attente[station] := paquets_en_attente[station]
            - 1;
            {  s'il reste des paquets en attente, on va émettre un
            nouveau délai pour le paquet en attente suivant }

            if paquets_en_attente[station] > 0 then
            begin
                prochain_cycle_pour_emettre[station] := cycle_courant
                +
                                                        random(m
                                                        ax_aleat)
                                                        + 1;
            end {if}
        end {if}
    end {if}
end; {procedure}



{**************************************************}


procedure  cumuler_paquets_en_attente;

    {  Ce module cumule le nombre total de paquets en attente sur
    l'ensemble des cycles de la simulation. En divisant le nombre
    obtenu moins le nombrede paquets en attente à la fin de la
    simulation par le nombre de cycles(duree_simulation), on
```

```
    obtiendra pour chaque station le nombre moyen de paquets en
    attente

    }

var
    nostation:  integer;

begin
    for nostation := 1 to NBSTATION do
        nombre_total_paquets_en_attente[nostation]
                :=      nombre_total_paquets_en_attente[nostation]
    +  paquets_en_attente[nostation];  {endfor}

end; {procedure}



{*************************************************}

procedure  imprimer_statistiques;


{  Cette procedure procède à l'impression des résultats à l'écran.
    Les résultats imprimés sont:
    le nombre de cycles de la simulation la valeur de max_aleat les
    probabilités de génération de paquets pour chaque station le
    nombre maximum et moyen de paquets en attente à chaque station
    le nombre de paquets généré par chaque station le rapport
    d'efficacité P / P+C, pour chaque station. le pourcentage
    d'intervalles  de  collision
    La valeur de "P" est calculée en retirant les paquets toujours en
    attente  à  la  fin  de  la  simulation
}


var
    ligcour,  nostation:  integer;

begin
```

```
ClrScr;
GotoXY(30,1);
writeln('S T A T I S T I Q U E S');
GotoXY(10,23);
writeln('Tapez <Shift>+<PrtScr> pour acheminer à l''imprimante');
GotoXY(10,5);
write('Durée de la simulation: ');
GotoXY(45,5);
writeln(duree_simulation);
GotoXY(10,6);
write('Longueur max de l''attente: ');
GotoXY(45,6);
writeln(max_aleat);
GotoXY(10,9);
writeln('STN Prob. Total Total Moyenne Collisions Efficacité (%)');
GotoXY(10,10);
writeln('# émis en attente en attente P / P + C ');
for nostation := 1 to NBSTATION do
begin
    GotoXY(10,11+nostation);
    write  (nostation);
    GotoXY(14,11+nostation);
    write  (prob_emission[nostation]:5:3);
    GotoXY(21,11+nostation);
    write  (nombre_paquets_generes[nostation]);
    GotoXY(28,11+nostation);
    write  (nombre_max_paquets_en_attente[nostation]);
    GotoXY(38,11+nostation);
    write  ((nombre_total_paquets_en_attente[nostation]
    -  paquets_en_attente[nostation])/  duree_simulation:7:1);
    GotoXY(52,11+nostation);
    write  (nombre_de_collisions[nostation]);
    GotoXY(60,11+nostation);
    write  ((nombre_paquets_generes[nostation]  -
    paquets_en_attente[nostation])/
        (nombre_paquets_generes[nostation]  -
        paquets_en_attente[nostation]  +
        nombre_de_collisions[nostation])  *  100:7:1);
end; {for}
GotoXY(10,11+NBSTATION+2);
writeln ('Nombre total de cycles avec collision: ',
```

```
                nombre_total_cycles_avec_collision);
    GotoXY(0,24);
end; {procedure}


{*********************************************

programme  principal
```

Programme: Simulation d'un bus avec NBSTATIONS stations. La discipline ressemble à un CSMA/CD où les collisions sont gérées par délai aléatoire

Description: Ce programme simule un bus sur lequel quatre stations émettent des paquets avec une probabilité déterminée. Lorsqu'une collision se produit, une procédure de traitement des collisions choisit un nombre au hasard entre 1 et max_aleat, lu en entrée. Ce nombre détermine le nombre de cycles pendant lesquels la station devra attendre avant de tenter une réémission. Des paquets peuvent être générés par les stations pendant que celle-ci attend pour une réémission. On considérera que ces paquets peuvent être tamponnés avec une capacité de tamponnage infinie. Des statistiques sont données à la fin de l'exécution. On a les paramètres d'entrée, le nombre total de cycles en collision et, pour chaque station, le nombre total de paquets émis, le nombre total de paquets en attente (un paquet sera compté n fois s'il a passé n cycles en attente), le nombre moyen de paquets en attente, le nombre de collisions pour cette station et une mesure d'efficacité tenant compte du nombre de paquets émis et du nombre de collisions. Pour cette dernière mesure, on a retranché les paquets encore en attente à la fin de la simulation. Une fonction "Randomize" a été utilisée afin de générer des suites pseudo-aléatoires différentes à chaque exécution.

```
                Limites: La saisie des données n'est aucunement
                validée. Il est de la responsabilité de l'utilisateur de
                faire attention aux valeurs entrées pour que celles-ci
                soient du bon type et d'une étendue correcte.
                Lasimulation ne devrait pas dépasser 30000 cycles, et
                ne pas être inférieure à 100 cycles, pour éviter des div.
                par 0,  les probabilités associées aux stations doivent
                être non-nulles. Si ces dernières dépassent 1, on aura
                l'équivalent de 1.
                Le nombre de cycles maximum d'attente ne doit pas
                dépasser la durée de la simulation et doit être
                supérieur à 0.L'algorithme d'implantation des
                probabilités utilise toujours le spectre inférieur de la
                distribution et il semble que le nombre de paquets
                générés dépasse presque toujours un peu la probabilité
                visée. Ce problème semble être dû à la fonction Random,
                qui semble n'être pas parfaitement uniforme.

Programmeur:    Hassan El-Moussaoui

Avril  1990

*************************************************}

begin

   lire_params;                    { lecture des paramètres }
   calcule_intervalle;             { préparation de l'implantation
                                   des prob. }
   init_simulation;                { initialisation des variables }


   { simulation proprement dite }

   for cycle_courant := 1 to duree_simulation do
   begin
      generer_nouveaux_paquets;          { on définit les
                                         nouveaux paquets }
      determiner_stations_autorisees;    { on détermine les
                                         stations autor. qui
                                         ont un pqt à émettre }
```

```
        cumuler_paquets_en_attente; { on cumule les paquets en
                                                    attente }

        if  situation_de_collision  then
            traiter_collision                { traitement de collision }
        else
            traiter_emission;                 { traitement d'émission unique
            }
        {endif}
    end;

    imprimer_statistiques                    { impression des statistiques }
end.
```

ANNEXE 2

```
(*************************************************)
(*                                               *)
(*    Programme:      Le logiciel de communication *)
(*                    a été développé pour       *)
(*                    gérer l'ensemble des capteurs *)
(*                    intelligents raccordés au réseau, *)
(*                    en appliquant le protocole de selection *)
(*                    décrit dans la mémoire.    *)
(*    Description:    Ce logiciel est présenté à l'opérateur *)
(*                    sous forme d'un menu déroulant, qui *)
(*                    contient des rubriques du menu *)
(*                    principal qui correspondent à des *)
(*                    fenêtres, ainsi que les rubriques *)
(*                    que ces fenêtres contiennent. *)
(*                    Pour ce faire on exécute le *)
(*                    programme principal et il va apparaître *)
(*                    sur l'ecran un menu déroulant dans *)
(*                    lequel on trouve cinq fenêtres. En *)
(*                    choisissant la fenêtre traitement, *)
(*                    dans laquelle il faut donner quelques *)
(*                    informations au système:   *)
(*                    - le jour et la date de l'expérience *)
(*                    sont données par le micro-ordinateur *)
(*                    - il faut entrer le numéro de l'expérience *)
(*                    - entrer aussi le nombre de capteurs *)
(*                    connectés sur le lien de communication *)
(*                    - l'heure du commencement de l'expérience *)
 (*                   sera donné par le systéme  *)
(*                    - il faut donné l'heure d'arrêt du système, *)
(*                    sans jamais dépasser 24 heures *)
(*                    - enfin on choisit une période *)
(*                    d'échantillonnage pour la prise des données *)
(*                    en minutes                 *)
(*    Date:           12 mai 1991                *)
(*    Auteur:         Hassan El-Moussaoui        *)
(*************************************************)
```

```
Program menuDeroulant;

Uses
      Crt, Dos, Graph, Rs232,
      t_decel,  t_verif,  t_fenetr,t_io;
      (*******************************************)
      (* Le programme utilise des unités du livre "Le grand livre *)
      (* du turbo pascal 6.0" d'édition Micro Application.        *)
            (* Ces unites serent à gérer l'écran ainsi pour les fonction
*)          (*  entrées/sorties.
*)
      (*******************************************)


{------------------ Déclaration des Constante-------------}
Const
  NbPointMax = 300;
  NbCapteurMax = 5;
  CouCadreExt = White;


{-------------------------- Déclaration des Types ----------}
Type
  Vecteur   = Array[1..NbPointMax] Of Real;
  Fenetre   = Record
                  TitreX,TitreY,Titre            : String;
                  VectX,VectY                    : Vecteur;
                  NbPoint                        : Integer;
                  CouCadre,CouGraph, CouGrad     : Integer;
                  CouTitre,CouTitreXY            : Integer;
                End;
  PetitePlanche = Array[1..3] Of Fenetre;


{------------------------ Déclaration des Variables ----------}
Var
   Fichier : Array[1..4] Of Text;
   NExperience, NCapteur, NbCapteur       : Integer;

   {----- Déclaration des Variables graphiques -----}
```

```
    Planche1                                    : PetitePlanche;
    Planche2, Planche3, Planche4                : Fenetre;
    dTemps, Moyen                               : Real;
    s,date                                      : string;
    principal                                   : t_barreMenDeroul;
    fenExist                                    : t_fenDispon;
    sousfen                                     : t_sousMenu;
    result, timer                               : integer;
    ancienFen                                   : t_pointImage;
    nom1, nom2, nom3, nom4                      : t_workstring;
    Entete                                      : text;


{---------------------------------------}

{$I  Graphiq.inc}

{---------------------------------------}

Function IntToStr(N:Integer)       :String;
Var
     s                  : String;
Begin
     Str(N,s); IntToStr := s;
End;

{-------------------------   La Moyenne ----------------}

Function MoyenneT(VectX,VectY   :Vecteur; Dim:Integer)       :Real;
Var
      i,k,j              : Integer;
     Somme               : Real;
     Change              : Boolean;

Begin
     Somme := 0;
     Change := False;
     k := 0;
     While (Not Change) And (k < Dim-1) Do
     Begin
          Inc(k); Change := (Vectx[k] > Vectx[k+1]);
```

```
        End;

        If Change Then For j := k+1 To Dim do
          Vectx[j] := Vectx[j] + 24;

        For i := 1 To Dim-1 Do
        Somme := somme + (VectX[i+1]-VectX[i])*(VectY[i+1] +
        VectY[i])/2; MoyenneT := Somme/(VectX[Dim]-VectX[1]);
End;


{--------------------------    AssignFichier    ----------------}

Procedure  AssignFichier(NExperience,NCapteur:Integer);
Begin
        Assign(Fichier[1],'ST_E'+IntToStr(NExperience)
                                +'C'+IntToStr(NCapteur)+'.dat');

        Assign(Fichier[2],'TP_E'+IntToStr(NExperience)
                                +'C'+IntToStr(NCapteur)+'.dat');

        Assign(Fichier[3],'PR_E'+IntToStr(NExperience)
                                +'C'+IntToStr(NCapteur)+'.dat');

        Assign(Fichier[4],'TM_E'+IntToStr(NExperience)
                                +'C'+IntToStr(NCapteur)+'.dat');

End;


{--------------------------    CloseFichier    ----------------}
Procedure  CloseFichier;
 Var
          i : Integer;
Begin
        For i := 1 To 4 Do
          Close(Fichier[i]);
End;
```

```
{--------------------------    ResetFichier    ------------------}
Procedure  ResetFichier;
Var
          i : Integer;
Begin
       For i := 1 To 4 Do
         Reset(Fichier[i]);
End;


{--------------------------    ReWriteFichier    ---------------}
Procedure  ReWriteFichier;
Var
          i : Integer;
Begin
       For i := 1 To 4 Do
         ReWrite(Fichier[i]); End;


{--------------------------    LireFichier    -------------------}
procedure  LireFichier;
Var
          i, N : Integer;
Begin
       ResetFichier;
       For i:= 1 To 3 Do
          With Planche1[i] Do
       Begin
             N := 0;
             While Not(EOF(Fichier[i])) Do
             Begin
                   Inc(N);  Read(Fichier[i],VectY[N]);
             End;
             NbPoint := N;
       End;
       N := 0;
       While Not(EOF(Fichier[4])) Do
       Begin
             Inc(N);
             With Planche1[1] Do
               Read(Fichier[4],VectX[N]);
```

```
        End;

        With Planche1[1] Do
        Begin
                Moyen := MoyenneT(VectX,VectY,N);
                For i := 1 To N Do
                Begin
                        VectX[i] := Frac(VectX[i])*10/6 + Trunc(VectX[i]);
                        Planche1[2].VectX[i] := VectX[i];
                        Planche1[3].VectX[i] := VectX[i];
                End;
        End;
        CloseFichier;
End;

{--------------------    Representer    -----------------}
Procedure Representer;
Var
          i : Integer;
Begin
        new(ancienFen);
        t_ecranVersHeap(ancienFen);
        t_creeFenetre(20,10,60,18,true,7);
        repeat
                t_clean(2,2,30);
                t_clean(2,6,25);
                gotoxy(2,2);
                write('Numéro de l''expérience : ');
                readln(NExperience);
                gotoxy(2,6);
                write('Nombre de capteurs : ');
                readln(NCapteur);
                nom1:='d:\tp\ST_E'+IntToStr(NExperience)
                                        +'C'+IntToStr(NCapteur)+'.dat';
                nom2:='d:\tp\PR_E'+IntToStr(NExperience)
                                        +'C'+IntToStr(NCapteur)+'.dat';
                nom3:='d:\tp\TP_E'+IntToStr(NExperience)
                                        +'C'+IntToStr(NCapteur)+'.dat';
                nom4:='d:\tp\TM_E'+IntToStr(NExperience)
                                        +'C'+IntToStr(NCapteur)+'.dat';
        until  t_fichier_exist(nom1)  and  t_fichier_exist(nom2)
```

```
                                    and  t_fichier_exist(nom3)
                                    and  t_fichier_exist(nom4);
        t_HeapVersEcran(ancienFen,true);
        For i := 1 To NCapteur Do
        Begin
              AssignFichier(NExperience,NCapteur);  LireFichier;
              InitialiseDriver;  ClearViewPort;  InitGraphe;
              PetitGraph(Planche1);
        End;
End;

{----------------------    RepresenterG    ----------}
Procedure  RepresenterG;
Var
        i ,j : Integer;
Begin
        new(ancienFen);
        t_ecranVersHeap(ancienFen);
        t_creeFenetre(20,10,60,18,true,7);
        repeat
              t_clean(2,2,30);
              t_clean(2,6,25);
              gotoxy(2,2);
              write('Numéro de l''expérience : ');
              readln(NExperience);
              gotoxy(2,6);
              write('Nombre de capteurs : ');
              readln(NCapteur);
              nom1:='d:\tp\ST_E'+IntToStr(NExperience)
                                  +'C'+IntToStr(NCapteur)+'.dat';
              nom2:='d:\tp\PR_E'+IntToStr(NExperience)
                                  +'C'+IntToStr(NCapteur)+'.dat';
              nom3:='d:\tp\TP_E'+IntToStr(NExperience)
                                   +'C'+IntToStr(NCapteur)+'.dat';
              nom4:='d:\tp\TM_E'+IntToStr(NExperience)
                                   +'C'+IntToStr(NCapteur)+'.dat';
        until  t_fichier_exist(nom1)  and  t_fichier_exist(nom2)
                                    and  t_fichier_exist(nom3)
                                    and  t_fichier_exist(nom4);
        t_HeapVersEcran(ancienFen,true);
        For i := 1 To NCapteur Do
```

```
        Begin
                AssignFichier(NExperience,NCapteur);
                LireFichier;
                InitialiseDriver;
                ClearViewPort;
                InitGraphe;
                For j := 1 To 3 Do
                Begin
                        ClearViewPort;
                        GrandGraph(Planche1[j]);
                        Repeat Until Ord(ReadKey) = 27;
                End;
        End;
End;

{---------------------------    Acquisition -----------------}
Procedure  Acquisition;
Type
      T_Str8 = String[8];
Var
    Capteur                         :Array[1..NbCapteurMax,1..4] Of Vecteur;
    i,n,j,l, nbr,code1, code2, code3,diff       : integer;
    c,d                            : char;
    x                              : word;
    result                           : array[1..255] of string[15];
    s,date                         : string;
    time, HeurIni, HeurFin         : t_str8;
    err                            : byte;
    delta, dtemps                   : integer;
    conver1, conver2, conver3 : longint;
    dheure, ttemps                  : real;



{---------------------------    ConvHeure    -----------------}
procedure ConvHeure(temps : longint ; var converh : real);
var
  partienti          : integer;
  partideci          : real;
begin
    partienti  :=  trunc(temps/3600);
```

```
    partideci := frac(temps/3600);
    converh := partienti + 60*partideci/100;
end;

{--------------------------  RepeteDon  ------------------}
procedure RepeteDon;
begin
     repeat
        gotoxy(6,5);
        write('l''heure de debut : ',HeurIni);
        t_clean(6,6,41);
        gotoxy(6,6);
        write('l''heure de fin (HH:MM:SS): ');
        readln(HeurFin);
        delta := T_sec_delta(HeurIni,HeurFin,err);
     until (err = 0);
        gotoxy(6,7);
        write('Entrez la periode d''echantillonnage en(min) = ');
        readln(dtemps);
end;

Begin
       new(ancienFen);
       t_ecranVersHeap(ancienFen);
       t_creeFenetre(8,7,75,25,true,5);
       t_ecrit(1,1,'la date: ');
       write(t_jour,',le  ',t_date);
       t_ecrit_clign(50,1,'TRAITEMENT');
       gotoxy(6,3);
       write('Numéro de l''expérience : ');
       readln(NExperience);
       gotoxy(6,4);
       write('Nombre de capteurs : ');
       readln(NbCapteur);
       time := t_heure(true,false);
       HeurIni := time;
       RepeteDon;
       t_HeapVersEcran(ancienFen,true);
       convert(Heurfin,conver2);
```

```
time  :=  t_heure(true,false);
Assign(Entete,'Entete'+IntTostr(Nexperience)+'.dat');
ReWrite(Entete);
WriteIn(Entete,NExperience);
Date := T_jour+' Le '+T_Date;
WriteIn(Entete,Date);
Close(Entete);
nbr := 0; x := 1;
InitialiseDriver;
repeat
        time  :=  t_heure(true,false);
        gotoxy(6,9);
        write(time);
        convert(time,conver1);
        if (conver1 mod (dtemps*60) = 0) then
        begin
              inc(nbr);
              For i := 1 To NbCapteur Do
              Begin
              repeat
                    result[x] := '';
                    setrs232On(1);
                    set_baud(300);
                    s  :=  IntToStr(i);
                    envoi(s[1]);
                    repeat
                          timer := 0;
                          repeat
                                inc(timer);
                                delay(1);
                          until donnee_prete or (timer = 5000);
                          if timer = 5000 then
                          begin
                                t_ecrit_clign(4,11,'ERREUR
                                     DANS LA TRANSMISSION');
                                Repeat
                                t_bip(3);
                                Until Ord(Readkey) <> 0;
                                exit;
                          end;
                          if donnee_prete then
```

```
                begin
                    d:= prend_caractere;
                    if d <> #70 then
                    result[x] := result[x] + d;
            end;
            until d = #70; Rs232Off;
            val(copy(result[x],1,5),Capteur[i,1][x],code1);
            val(copy(result[x],6,5),Capteur[i,2][x],code2);
            val(copy(result[x],11,5),Capteur[i,3][x],code3
            );
until (code1 = 0) and (code2 = 0) and (code3 = 0);
End;

time  := t_heure(true,false);
convert(time,conver1);
convheure(conver1,Capteur[i,4][x]);
Planche1[1].VectX[x] := Capteur[i,4][x];
With Planche1[1] DO
VectX[x] := Frac(VectX[x])*10/6 + Trunc(VectX[x]);
Planche1[2].VectX[x] := Planche1[1].VectX[x];
Planche1[3].VectX[x] := Planche1[1].VectX[x];
Planche1[1].VectY[x] := Capteur[i,1][x];
Planche1[2].VectY[x] := Capteur[i,2][x];
Planche1[3].VectY[x] := Capteur[i,3][x];
If x > 1 Then
Begin
      Planche1[1].NbPoint := x;
      Planche1[2].NbPoint := x;
      Planche1[3].NbPoint := x;
      ClearViewPort;
      InitGraphe;
      PetitGraph(Planche1);
      GoToXY(5,3);
      Write('Styrene:',capteur[i,1][x]:6:2);
      GoToXY(5,4);
      Write('Temperature:',Capteur[i,2][x]:6:2);
      GoToXY(5,5);
      Write('Pression:',Capteur[i,3][x]:6:2);
      GotoXY(5,6);
      Write('Moyenne:',MoyenneT(Capteur[i,4],
                                Capteur[i,1],x):6:2);
```

```
                End;
                inc(x);
            end;
            time  :=  t_heure(true,false);
            convert(time,conver1);
            diff  :=  abs(conver1  -  conver2);
        until   t_int_range(0,10,diff);
        CloseGraph;
        For i := 1 To NbCapteur Do
        Begin
            AssignFichier(NExperience,i);
            ReWriteFichier;
            For j := 1 To nbr Do
            Begin
                WriteLn(Fichier[1],Capteur[i,1][j]);
                WriteLn(Fichier[2],Capteur[i,2][j]);
                WriteLn(Fichier[3],Capteur[i,3][j]);
                WriteLn(Fichier[4],Capteur[i,4][j]);
            End;
            CloseFichier;
        End;
End; { Fin Acquisition }

{--------------------------     MENU     -------------------------}
procedure  initMenu;
const
   ligne   =   3;
begin
    principal[1].design     := 'Traitement ';
    principal[1].col        := 4;
    principal[1].lign       := ligne;

    principal[2].design   := '  Resultats   ';
    principal[2].col        := 22;
    principal[2].lign       := ligne;

    principal[3].design   := 'Date et heure ';
    principal[3].col        := 45;
    principal[3].lign       := ligne;
    principal[4].design   := 'Quit ';
    principal[4].col        := 68;
```

```
    principal[4].lign       := ligne;

    fenExist[1] := false;
    fenExist[2] := true;
    fenExist[3] := true;
    fenExist[4] := false;

    sousFen[2].colHG      := 20;
    sousFen[2].lignHG     := ligne+4;
    sousFen[2].design[1] := 'Petite Courbes ';
    sousFen[2].design[2] := 'Grande Courbes ';
    sousFen[2].nombre     := 2;
    sousFen[2].longMax    :=15;

    sousFen[3].colHG      := 43;
    sousFen[3].lignHG     : = ligne+4;
    sousFen[3].design[1] := '     date      ';
    sousFen[3].design[2] := '     Heure     ';
    sousFen[3].nombre     := 2;
    sousFen[3].longMax    :=15;
end;


{--------------------  Programme Principale  ---------}
Begin
      initMenu;
      repeat
        t_cadre(2,1,79,5,true,true,true);
        t_curseurInvis; result :=
        t_menuDeroulant(principal,4,1,true,fenExist,sousFen,true,5);
        t_curseurVisibl;
        case result of
              0       :   begin
                                t_ecrit(5,15,'Fin');
                                t_attendre;
                                t_clean(5,15,40);
                          end;

              1       :   begin
                                Acquisition;
                          end;
```

```
21      :   begin
                Representer;
                Assign(Entete,'Entete'
                      +IntTostr(NXperience)+'.dat');
                Reset(Entete);
                ReadIn(Entete,NExperience);
                Gotoxy(2,2);
                Write('Experience : ',Nexperience);
                ReadIn(Entete,Date);
                Gotoxy(2,3);
                Write('Date : ',Date);
                Gotoxy(2,4);
                Write('La moyenne : ',moyen:6:2,'
                      ppm');
                Repeat Until Ord(Readkey) = 27;
                  CloseGraph;
            end;

22 :        begin
                RepresenterG;
                CloseGraph;
            end;

31      :   begin
                t_ecrit(5,15,'la date: ');
                write(t_jour,',le  ',t_date);
                t_attendre;
                t_clean(5,15,40);
            end;

32      :   begin
                t_ecrit(5,15,'la date: ');
                write(t_jour,',le  ',t_date);
                t_attendre;
                t_clean(5,15,40);
            end;

4       :   begin
                ClrScr; Halt(0);
            end;
```

```
        end;
        until (result = 0) or (result = 4);
End.



(*************************************************)
(*  Unité :  rs232.pas                           *)
(*  Date   :  06/11/1990                         *)
(*  Compiler :  Turbo pascal 6.0                 *)
(*  Auteur  :  Hassan El-Moussaoui               *)

(*  Cette unité contient des routines qui servent à gerer    *)
(*  l'inteface série ACE-8250, et des routines de reception  *)
(*  et de transmission des caractères                        *)
(*************************************************)
unit  rs232;

interface

uses        dos, crt;

var
     donnee_prete:boolean;
     baud_rate:longint;
     dimension_de_la_Queue:word;
     function  Prend_caractere:char;
     procedure envoi(carac:char); (* transmet un caractere *)
     procedure  envoi_phrase(phrase:string);
                              (* transmet un caractere *)
     procedure SetRS232ON(COM:byte);
     procedure  Set_baud(baud:integer);
     procedure Purge_Queue;
     procedure  Rs232Off;
     Procedure Break;

implementation

const (* registres du 8250 *)
```

```
      LCR_1       =     $3FB;
      MCR_1       =     $3FC;
      DLAB_1      =     $3F8;
      LSR_1       =     $3FD;
      MSR_1       =     $3FE;
      TXB_1       =     $3F8;
      RXB_1       =     $3F8;
      IE_1        =     $3F9;
      LCR_2       =     $2FB;
      MCR_2       =     $2FC;
      DLAB_2      =     $2F8;
      LSR_2       =     $2FD;
      MSR_2       =     $2FE;
      TXB_2       =     $2F8;
      RXB_2       =     $2F8;
      IE_2        =     $2F9;
      ESC         =     ^[;
      Nul         =     ^@;

 var
      LCR,MCR,DLAB,LSR,MSR,TXB,RXB,IE        :integer;
      buffer      :array[0..255]ofbyte;
      Queue_entree: integer; Queue_sortie     :
      integer;
       interruption     : pointer;
            CARAC              :byte;
            Xon                :boolean;
            DejaInit           :word;


      procedure Rs232Off;
      begin
              Case dejaInit of
                11:port[$21] := port[$21] or $08;
                12:port[$21] := port[$21] or $10;
              end;
              if dejaInit > 0 then setintvec(dejainit,interruption);
end;


      (*************************************************)
```

```
        (* Ceci c'est la routine d'interruption en tant que tel *)

(*************************************************)


procedureRecoi_dans_Queue(Flags,CS,IP,AX,BX,CX,DX,SI,DI,DS,ES,BP
                                        :word); interrupt;
begin
if (port[LSR] and 1) > 0 then begin
                    CARAC:=port[RXB];
                    If Carac = 19 then XON:=FALSE else XON:=True;
                    Buffer[Queue_entree]:=Carac;
                    Queue_entree := (Queue_entree+1) and $FF;
                    inc(dimension_de_la_queue);
                    Donnee_prete:=true;
end; { Gestion du EOI sur le 8259 }
case DejaInit of
          11:port[$20]:=$63;
          12:port[$20]:=$64;
end;
end;



(*************************************************)

 (* Routine de gestion de la reception en mode serie *)

(*************************************************)

Procedure Set_Rs232_com_1;
 var
    rien:byte;


(*************************************************)
(*Cette partie initialise le processus d'interruption*)
 (*************************************************)
```

```
begin
     RS232Off;
     getintvec(12,interruption)
     ;
     setintvec(12,@recoi_dans_Queue);
     (*assure la vidange du port d'entree*)
     rien:=port[RXB_1];rien:=port[RXB_1];
     rien:=port[RXB_1];rien:=port[RXB_1]; (*assure le reset des
                                            registres de controle de
                                            ligne et modem*)
     rien:=port[LSR_1];
     rien:=port[MSR_1];
     donnee_prete:=false;
      Queue_entree:=0;
      Queue_sortie:=0;
      XON:=true;
      LCR:=LCR_1;
      MCR:=MCR_1;
      DLAB:=DLAB_1;
      LSR:=LSR_1;
      MSR:=MSR_1;
      TXB:=TXB_1;
      RXB:=RXB_1; I
      E:=IE_1;
      DejaInit:=12;
      port[$21]:=port[$21]and  $ef;
end;

Procedure  Set_Rs232_com_2;
var
   rien:byte;


(*********************************************)

 (*  Cette partie initialise le processus d'interruption *)

     (*******************************************)
```

```
begin
     Rs232Off;
     getintvec(11,interruption);
     setintvec(11,@recoi_dans_Queue); (*assure la vidange du port
                                              d'entree*)
     rien:=port[RXB_2];
     rien:=port[RXB_2];
     rien:=port[RXB_2];
     rien:=port[RXB_2]; (*assure le reset des registres de controle de
                              ligne et modem*)
     rien:=port[LSR_2];
     rien:=port[MSR_2];
     donnee_prete:=false;
     Queue_entree:=0;
      Queue_sortie:=0;
      XON:=true;
      LCR:=LCR_2;
      MCR:=MCR_2;
      DLAB:=DLAB_2;
      LSR:=LSR_2;
      MSR:=MSR_2;
      TXB:=TXB_2;
      RXB:=RXB_2;
      IE:=IE_2;
      DejaInit:=11;
      port[$21]:=port[$21] and $F7;
end;


(*************************************************)
(*                  Procedure initialisant le port serie       *)
(*                entrer les donnees comme suit                *)
(*                     set_baud(baudrate);                     *)
(*************************************************)
procedure  set_baud(baud:integer);
var
      valeur:longint;
begin
    valeur := 115200 div baud;
    baud_rate:= 115200 div valeur;
```

```
    port[LCR]:=128; (*accede le divisor latch*)
    portW[DLAB]:=valeur; (*set le baud rate*)
    port[LCR]:=3; (*8 bits,1stop,pas de parité , pas de break*)
    port[IE]:=1; (*set les interruptions*)
    port[MCR]:=9; (*port modem et interruptions*)
end;


(*************************************************)
(*          Procedure transmettant un caractere          *)
(*          ex :                                         *)
(*    envoi('C');        --->envoi un    C          *)
(*************************************************)



procedure envoi(carac:char); (* transmet un caractere *)
var
heure_debut:longint;
begin
         heure_debut:=0;
         repeat
                  inc(heure_debut); delay(1);
                  if heure_debut > 30000 then begin
                  XON := true;
       end;
         until XON;
               (* attend avant de transmettre que XON soit vrai *)
         repeat until ((port[LSR] and $20) = $20);
         port[TXB]:=ord(carac);
end;

procedure envoi_phrase(phrase:string); (* transmet un caractere *)
      var i:word;
           begin
                for i:=1 to length(phrase) do envoi(phrase[i]);
      end;

(*************************************************)
* fonction qui enleve un caractere de la queue de *)
(* réception par interruption *)
```

```
(* repeat until donnee_prete *)
(*car:=prend_caractere;*)
(************************************************)

function  Prend_caractere:char;
begin
      prend_caractere:=char(buffer[Queue_sortie]);
      Queue_sortie:= (Queue_sortie + 1) and $FF;
      if Queue_sortie = Queue_entree
      then  donnee_prete:=false;
      dec(dimension_de_la_queue);
end;




(************************************************)

       (* Procedure de vidange de la queue *)

(************************************************)

procedure Purge_Queue;
begin
      Queue_entree:= Queue_sortie;
      donnee_prete:=false;
end;

procedure SetRS232ON(COM:byte);
begin
      if dejaInit > 0 then setintvec(dejainit,interruption);
      case com of
            2     :     Set_Rs232_com_2;
            1     :     Set_Rs232_com_1;
      end;
end;

procedure Break;
var
      b     :     byte;
      begin
```

```
                b:=port[LCR] ;
                port[LCR]:=b or 64;
                delay(500);
                port[LCR]:=b;
end;

begin
        dejaInit:=0;  dimension_de_la_queue:=0;
end.
```

```
(*************************************************)
(*  Fichier inclu :     Graphiq.inc                *)
(*  Date          :     02  juin 1991              *)
(*  Auteur        :     Hassan El-Moussaoui        *)
(*  Ce fichier contient les fonctions et les procedures, qui  *)
(*  servent à réaliser les graphiques et les coubes des       *)
(*  résultats obtenus par le prigramme d'acquisition          *)
(*************************************************)

{----------------------  Minimum  d'un  vecteur  ---------------}

Function Min(Vect:Vecteur; Dim:Integer):Real;
Var i : Integer;
x : Real;
Begin
      x := 10E+32; For i := 1 To Dim Do
      If Vect[i] < x Then x := Vect[i]; Min := x;
End; { Min }

{-----------------------  Maximum  d'un  vecteur  --------------}

 Function Max(Vect:Vecteur; Dim:Integer):Real;
Var i : Integer;
x : Real;
Begin
      x :=-10E-32; For i := 1 To Dim Do
      If Vect[i] > x Then x := Vect[i]; Max := x;
End; { Max }

{----------------------  Moyenne  d'un  vecteur  --------------}

Function Moyenne(Vect:Vecteur; Dim:Integer):Real;
Var i : Integer;
x : Real;
Begin
      x := 0; For i := 1 To Dim Do
      x := x + Vect[i]; Moyenne := x/Dim;
End; { Moyenne }
```

```
{---------------------   10   puissance   P   -------------------}

Function  Puiss10(P:Integer):Real;
Var i : Integer;
X : Real;
Begin
      X := 1.0; If P > 0 Then For i := 1 To P Do
      X := X * 10 Else If P < 0 Then For i := P To 1 Do
      X := X / 10; Puiss10 := X;
End; { Puiss10 }

{-----------------------------------------------------------}
Procedure  InitialiseDriver;
var
  InGraphicsMode : boolean;
  PathToDriver  : string;
  GraphDriver, GraphMode, errorCode : Integer;
begin
  DirectVideo := False;
  PathToDriver := ''; repeat
{$IFDEF Use8514}
    GraphDriver := IBM8514;
    GraphMode := IBM8514Hi;
{$ELSE}
GraphDriver := Detect;
{$ENDIF}
    InitGraph(GraphDriver, GraphMode, PathToDriver);
    ErrorCode := GraphResult;
    if ErrorCode <> grOK then
    begin
      WriteIn('Erreur graphique : ', GraphErrorMsg(ErrorCode));
      if ErrorCode = grFileNotFound then
      begin
        WriteIn('Entrer le chemin pour le BGI driver :');
        ReadIn(PathToDriver);  WriteIn;
      end
      else
    Halt(1);
    end;
until ErrorCode = grOK;
```

```
end; { InitialiseDriver }

{-------------------------------------------------------------}
Procedure ScaleY(Vect:Vecteur; NbPoint:Integer; Var Grad,Centre:Real;
                        Var NbDecimal:Integer);
Var TempReel : Real;
    j, k     : Integer;

Begin
      Grad := (Max(Vect,NbPoint)-Min(Vect,NbPoint))/5; j := 0;
      While (Grad > 10) Do
       Begin
            Grad := Grad/10;
            Inc(j);
       End;
      If (Round(Grad) = Trunc(Grad)) Then
      Grad := (Round(Grad)+0.5)*Puiss10(j)
      Else
         Grad := Round(Grad) * Puiss10(j);
      Centre := (Min(Vect,NbPoint)+Grad/2+(Max(Vect,NbPoint)-
      Min(Vect,NbPoint))/2)/Puiss10(j) + 0.25;
      If (Round(Centre) <> Trunc(Centre)) Then
      Centre := (Round(Centre)-0.5)*Puiss10(j)
      Else
         Centre := Round(Centre) * Puiss10(j);
      NbDecimal := 0;
      TempReel := Grad;
      While ((TempReel - Trunc(TempReel)) <> 0.0) And (NbDecimal <
                                                          2) Do
      Begin
            TempReel := TempReel*10;
            Inc(NbDecimal);
      End;
End; { ScaleY }

{-------------------------------------------------------------}
Procedure ScaleX(Var Vect:Vecteur; NbPoint:Integer; Var
                                        Grad,Centre:Real);
Var j, k : Integer;
Change : Boolean;
Begin
```

```
        Change := False; k := 0;
        While (Not Change) And (k < NbPoint-1) Do
        Begin
                Inc(k);
                Change := (Vect[k] > Vect[k+1]);
        End;
        If Change Then
        For j := k+1 To NbPoint do
          Vect[j] := Vect[j] + 24;
        Grad := (Max(Vect,NbPoint)-Min(Vect,NbPoint))*100/5;
        j := 0;

        While (Grad > 10) Do
         Begin
                Grad := Grad/10;
                Inc(j);
        End;
        If (Round(Grad) = Trunc(Grad)) Then
        Grad := (Round(Grad)+0.5)*Puiss10(j)
        Else
            Grad := Round(Grad) * Puiss10(j);
        Centre :=
        (100*Min(Vect,NbPoint)+Grad/2+(100*Max(Vect,NbPoint)-
                    100*Min(Vect,NbPoint))/2)/Puiss10(j) + 0.25;
        If (Round(Centre) <> Trunc(Centre)) Then
        Centre := (Round(Centre)-0.5)*Puiss10(j)
        Else
        Centre := Round(Centre) * Puiss10(j);
        Centre := Centre/100;
        Grad := Grad/100;
        While (Grad < 0.01) Do
         Grad := Grad*2;
End; { ScaleX }

{-------------------------------------------------------------}
Procedure TraceGraph(Planche:Fenetre; X0,Y0,LongX,LongY:Integer);
Var
    GradX, GradY, CentreX, CentreY, TempReel : Real;
    j, NbDecimal, Haut : Integer;
    Msg : String;
Begin
```

```
With Planche Do
Begin
      SetColor(CouCadre);
      Rectangle(X0,Y0,X0+LongX,Y0-LongY);
      SetColor(CouGrad); For j := 1 To 4 do
      Begin
            Line(X0,Y0-Round((LongY/5)*j),X0+3,Y0-
                                          Round((LongY/5)*j));
            Line(X0+Round((LongX/5)*j),Y0,X0+Round((LongX/5)*
                                                j),Y0-3);
      End;
      SetColor(CouTitre);
      SetTextStyle(DefaultFont,HorizDir,1);
      SetTextJustify(CenterText,BottomText);
      MoveTo(X0+LongX Div 2,Y0-LongY-4);
      OutText(Titre);
      Haut := TextHeight('T');
      SetColor(CouTitreXY);
      SetTextStyle(DefaultFont,HorizDir,1);
      SetTextJustify(CenterText,BottomText);
      MoveTo(X0+LongX Div 2,Y0+2*Haut+11);
      OutText(TitreX);
      SetTextStyle(DefaultFont,VertDir,1);
      SetTextJustify(CenterText,CenterText);
      MoveTo(X0-TextWidth('00000')-13,Y0-(LongY Div 2));
      OutText(TitreY);
      ScaleY(VectY,NbPoint,GradY,CentreY,NbDecimal);
      SetColor(CouGrad);
      SetTextStyle(DefaultFont,HorizDir,1);
      SetTextJustify(RightText,CenterText); For j := 0 To 5 Do
      Begin
            TempReel := CentreY+(j-3)*GradY;
            If (Abs(TempReel*Puiss10(NbDecimal)) < 1.0) Then
            TempReel := Abs(TempReel);
            Str(TempReel:1:NbDecimal,Msg); MoveTo(X0-5,Y0-
                              Round((LongY/5)*j)); OutText(Msg);
      End;
      ScaleX(VectX,NbPoint,GradX,CentreX);
      SetColor(CouGrad);
      SetTextStyle(DefaultFont,HorizDir,1);
      SetTextJustify(CenterText,BottomText);
```

```
            For j := 0 To 5 do
            Begin
                  TempReel := CentreX+(j-3)*GradX;
                  If TempReel < 0 Then TempReel := TempReel + 24;
                  TempReel := Frac(TempReel)*6/10 +
                                                Trunc(TempReel);
                  If TempReel >= 24 Then TempReel := TempReel - 24;
                  If (Abs(TempReel*Puiss10(2)) < 1.0) Then
                  TempReel := Abs(TempReel);
                  Str(TempReel:1:2,Msg);
                  MoveTo(X0+Round((LongX/5)*j),Y0+Haut+4);
                  OutText(Msg);
            End;
            SetColor(CouGraph);
            For j := 2 To NbPoint Do
            Begin
                  Line(Round(X0+(LongX-2)*3/5+(VectX[j-1]-
                        CentreX)*(LongX-2)/(5*GradX)),
                  Round(Y0-(LongY-2)*3/5-(VectY[j-1]-
                  CentreY)*(LongY-2)/(5*GradY)), Round(X0+(LongX-
                        2)*3/5+(VectX[j]-CentreX)*(LongX-
                        2)/(5*GradX)), Round(Y0-(LongY-2)*3/5-
                        (VectY[j]- CentreY)*(LongY-2)/(5*GradY)))
            End;
      End;
End; { TraceGraph }

{-------------------------------------------------------------}
Procedure CadreExt(Var Haut1,Haut2:Integer);
Begin
      SetTextStyle(DefaultFont,HorizDir,2);
      SetTextJustify(CenterText,BottomText);
      Haut1 := 0;
      Haut2 := (GetMaxY+Haut1) Div 2;
      MoveTo(GetMaxX Div 2, Haut1-4);
      SetColor(CouCadreExt);
      Rectangle(0,0,GetMaxX,GetMaxY);
      Line(0,Haut1,GetMaxX,Haut1);
      SetTextStyle(DefaultFont,HorizDir,1);
End; { CadreExt }
```

```
{-------------------------------------------------------------}
Procedure  PetitGraph(Planche:PetitePlanche);
Var
      Haut1,Haut2,Haut, X0, Y0, LongX, LongY : Integer;
Begin
      CadreExt(Haut1,Haut2);
      Line(GetMaxX Div 2,Haut1,GetMaxX Div 2,GetMaxY);
      Line(0,Haut2,GetMaxX,Haut2);
      Haut := TextHeight('T');
      X0 := 5 + TextWidth('000000');
      Y0 := GetMaxY - 2*Haut - 15; LongX := GetMaxX Div 2 - X0 - 23;
      LongY := Haut2 - Haut1 - 3*Haut - 25;
      TraceGraph(Planche[1],X0,Y0,LongX,LongY);
      X0 := X0 + GetMaxX Div 2;
      TraceGraph(Planche[2],X0,Y0,LongX,LongY);
      Y0 := Haut2 - 2*Haut - 15;
      TraceGraph(Planche[3],X0,Y0,LongX,LongY);
End; { PetitGraph }

{-------------------------------------------------------------}
Procedure  GrandGraph(Planche:Fenetre);
Var
      Haut1,Haut2,Haut, X0, Y0, LongX, LongY : Integer;
Begin
      CadreExt(Haut1,Haut2);
      Haut := TextHeight('T');
      X0 := 5 + TextWidth('0000000');
      Y0 := GetMaxY - 2*Haut - 15;
      LongX := GetMaxX - X0 - 15;
       LongY := GetMaxY - Haut1 - 3*Haut - 25;
      TraceGraph(Planche,X0,Y0,LongX,LongY);
End; { GrandGraph }

{-------------------------------------------------------------}
procedure  InitGraphe;
Begin
     With  Planche1[1] Do
     Begin
            TitreX := 'Temps en heure (HH.MM)'; TitreY := ' ';
            Titre := 'Concentration de Styrene en ( ppm )';
            CouCadre := White;
```

```
            CouGraph := White;
            CouGrad := White;
            CouTitre := White;
            CouTitreXY := White;
        End;
        With Planche1[2] Do
        Begin
            TitreX := 'Temps en heure (HH.MM)';
            TitreY := ' ';
            Titre := 'Temperature en ( °C )';
            CouCadre := White;
            CouGraph := White;
            CouGrad := White;
            CouTitre := White;
            CouTitreXY := White;
        End;
        With Planche1[3] Do
        Begin
            TitreX := 'Temps en heure (HH.MM)';
            TitreY := ' ';
            Titre := 'Pression Relative en ( kpa )';
            CouCadre := White;
            CouGraph := White;
            CouGrad := White;
            CouTitre := White;
             CouTitreXY := White;
        End;
        With Planche2 Do
        Begin
            TitreX := 'TitreX3';
            TitreY := 'TitreY3';
              Titre := 'Titre3';
            CouCadre := White;
             CouGraph := 14;
             CouGrad := White;
            CouTitre := White;
            CouTitreXY := White;
      End;
End;
```

ANNEXE 3

```
*************************************************
*           Programme:  Ce programme est implanté au niveau du
*                       capteur intelligent (68HC11).  Pour
*                       engager une bonne communication,  il faut
*                       que ce programme soit compatible avec
*                       celui de la station central (micro-
*                       ordinateur).
*           Description: Lorsque les valeurs calculées, de
*                       concentration de styrènes, de température
*                       et de pression relative,  ils sont
*                       mémorisées à des adresses respectives, on
*                       teste si le capteur  choisi peut entrer en
*                       communication avec la station de contrôle.
*                       Ce test est vérifié par la consultation du
*                       registre de réception du M68HC11;  si
*                       l'adresse reçue correspond bien au capteur
*                       en question, on passe à la transmission
*                       des valeurs en mémoire.  Sinon on
*                       commence la même procédure depuis le
*                       début jusqu'à la réception de la bonne
*                       adresse.
*           Date:       06 mai 1991
*           Auteur:     Hassan El-Moussaoui
*************************************************

            ORG         $E000
            LDS         #$00FF   Initialisation de la pile
            LDAB        # 0      Registre de contrôle1 est mit à 0
            STAB        SCCR1
            LDAB        #$0C     Registre de contrôle2 est fixé à
                                 $0C de tel sorte, à ce qu'il soit
                                 en mode de transmission ou de
                                 réception.
            STAB        SCCR2
            LDAB        #$35     Vitesse de transmission et de
                                 recéption est reglée à 300 baud/s
            STAB        BAUD
```

```
**********************************************
*            Programme principal
**********************************************

TOUJOUR :   JSR     TEMPERA     Routines pour l'acquisition et
            JSR     PRESSIO     le calcul des valeurs de
            JSR     STYRENE     température, pression et la
                                concentration du styrène, et
                                de les palcer dans des
                                mémoires alouées pour
                                chaque valeur.  Ces routines
                                sont décrite dans le
                                programme d'acquisition
                                développé par Elhiri
                                Mohammed (Développement
                                de senseur intelligent pour la
                                détection du styrène)

            JSR     INPSCI      Routine qui test la réception d'un
                                caractère
            LDAA    SCDR
            ANDA    #$7F        Masqué le septiéme bit (ASCII)
            CMPA    #NUMERO     Comparez le numéro reçu avec
                                celui du capteur en question s'il
                                sont égales on procéde à la
                                transmission des données par la
                                routine TRANSC2.  Sinon on test
                                encore le registre de reception

            BNE     TOUJOUR
            JSR     TRANASC2
            JMP     TOUJOUR
```

```
*********************************************
*           Routine qui sert à la transmission de tous les données
*           stockées dans les memoires:
*               MEMOS (pour la valeur de la concentration de
*                       styrène)
*               MEMOT (pour la valeur de la température)
*               MEMOP (pour la valeur de la pression)
*********************************************
TRANASC2  EQU      *
          LDX      #MEMOS
          JSR      TRANASC1
          LDX      #MEMOT
          JSR      TRANASC1
          LDX      #MEMOP
          JSR      TRANASC1
          JSR      FIN
          RTS


*********************************************
*           Routine qui impose le format des messages à envoyer au
*           micro.  Ce format fait partie du protocole choisi pour la
*           communication mentionné à la séction (2.3).  Une fois
*           qu'on détecte un espace dans la valeur à envoyer on le
*           remplace par un zéro.
*********************************************
TRANASC1  EQU      *
          LDAA     0,X
          CMPA     #$30
          BNE      TRANS1
          JSR      ECRI0
TRANS1:   LDAB     #$05
ENCORE:   LDAA     0,X
          CMPA     #$20
          BNE      CONTINU
          LDAA     #$30
CONTINU:  JSR      TRANSMIS
          DECB
          BNE      ENCORE
          RTS
```

```
**********************************************
*           Routine qui écrit des zéros, ces dernier remplacent les
*           espaces dans les valeurs à transmettre.
**********************************************
ECRI0       EQU         *
            PSHX
            LDAB        #$30
            LDAA        #$04
ENCOR1      INX
            STAB        0,X
            DECA
            BNE         ENCOR1
            PULX
            RTS

**********************************************
*           Routine qui indiquent la fin de transmission des
*           messages.
**********************************************
FIN         EQU         *
            LDAA        #$46
            JSR         OUTSCI
            ANDA        #$7F
            STAA        SCDR
            JSR         DLY10
            RTS

**********************************************
*           Routine de transmission pour plusieurs caractères
**********************************************
TRANSMIS    EQU         *
            PSHB
            JSR         OUTSCI
            ANDA        #$7F
            STAA        SCDR
            JSR         DLY10
            INX
            PULB
            RTS
```

```
*************************************************
*           Routine qui test la réception d'un caractère
*************************************************
INPSCI      EQU         *
TEST        LDAB        SCSR
            BITB        #$20
            BEQ         TEST
            RTS

*************************************************
*           Routine qui test la transmission d'un caractére
*************************************************
OUTSCI      EQU         *
TEST1       LDAB        SCSR
            BITB        #$80
            BEQ         TEST1
            RTS

*************************************************
*           Routine qui calcul un delai
*************************************************
DLY10       EQU         *
            PSHX
            LDX         #$FFFF
DLOOP       DEX
            BNE         DLOOP
            PULX
            RTS

* *
END
```

ANNEXE 4

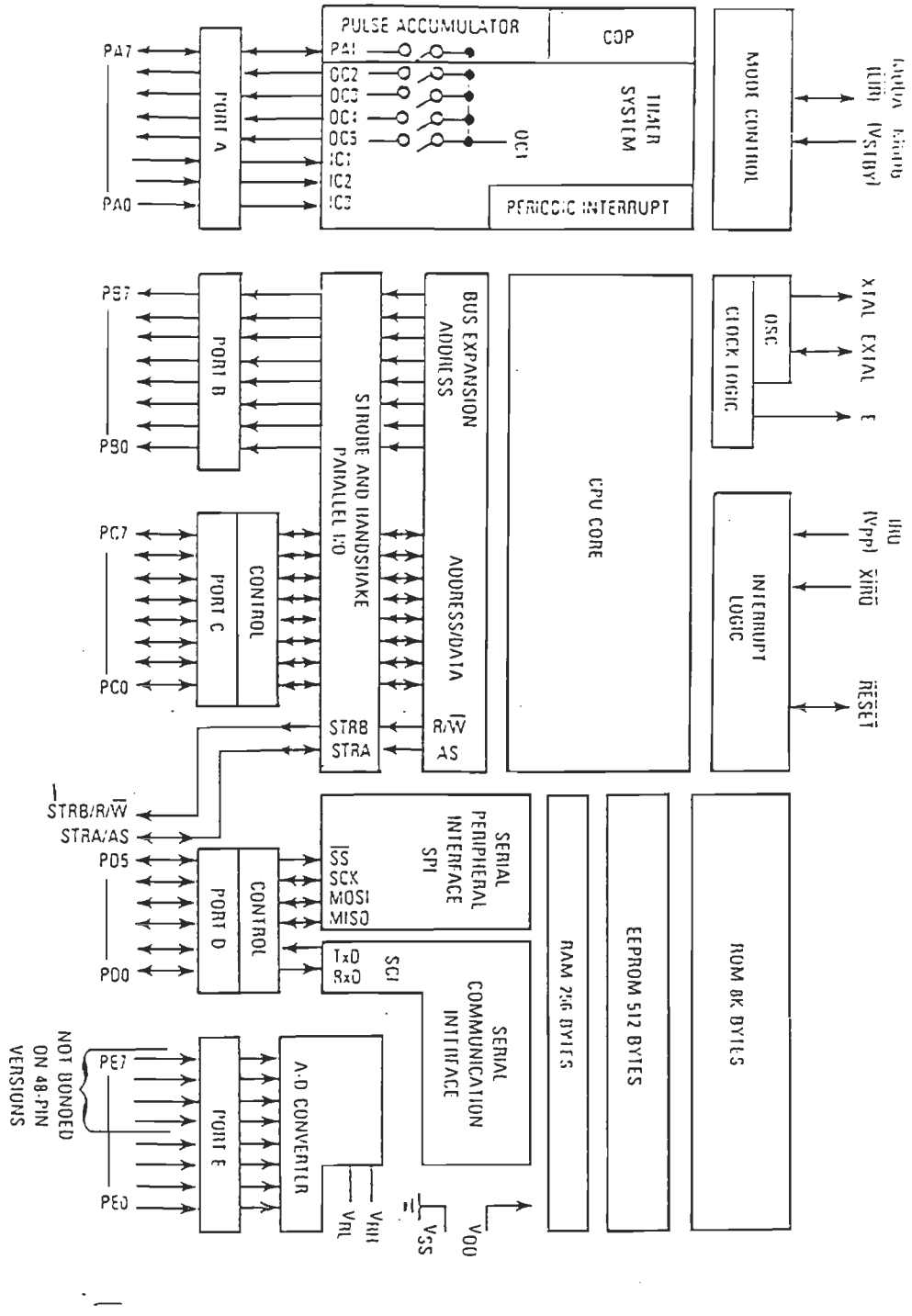


Schéma bloc du M68HC11

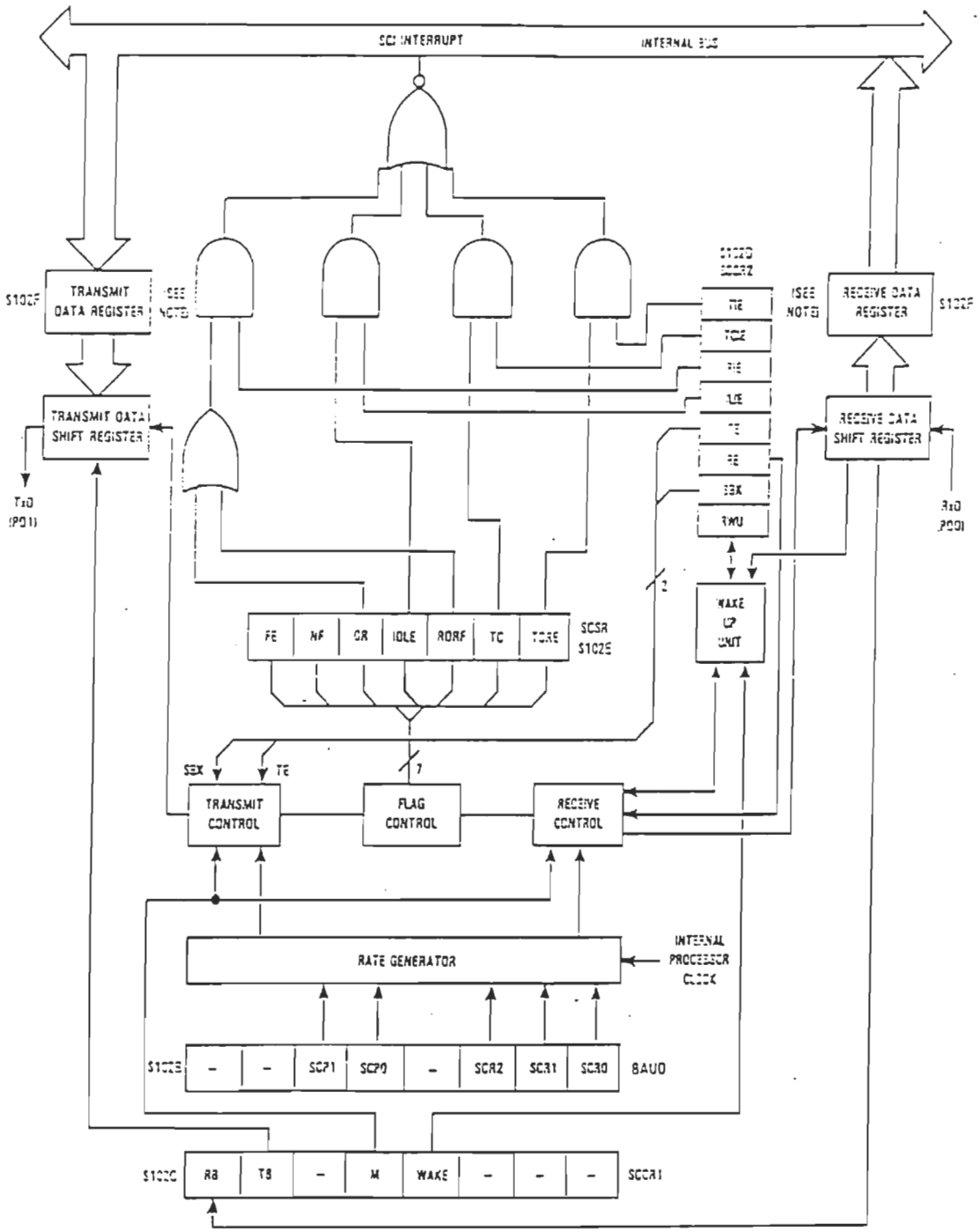


Schéma bloc de l'interface de communication d'un M68HC11

ANNEXE 5

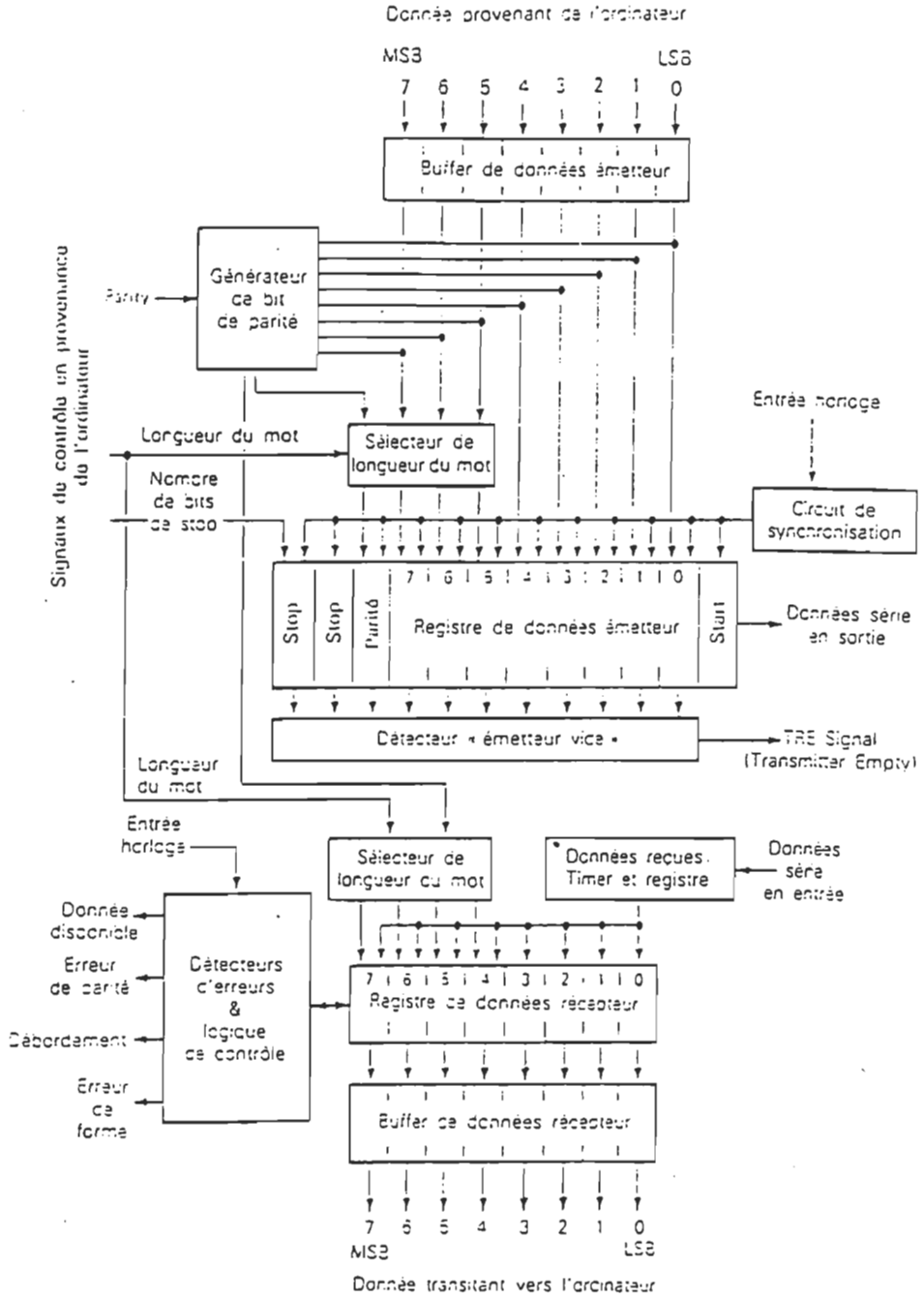


Schéma fonctionnel d'une UART

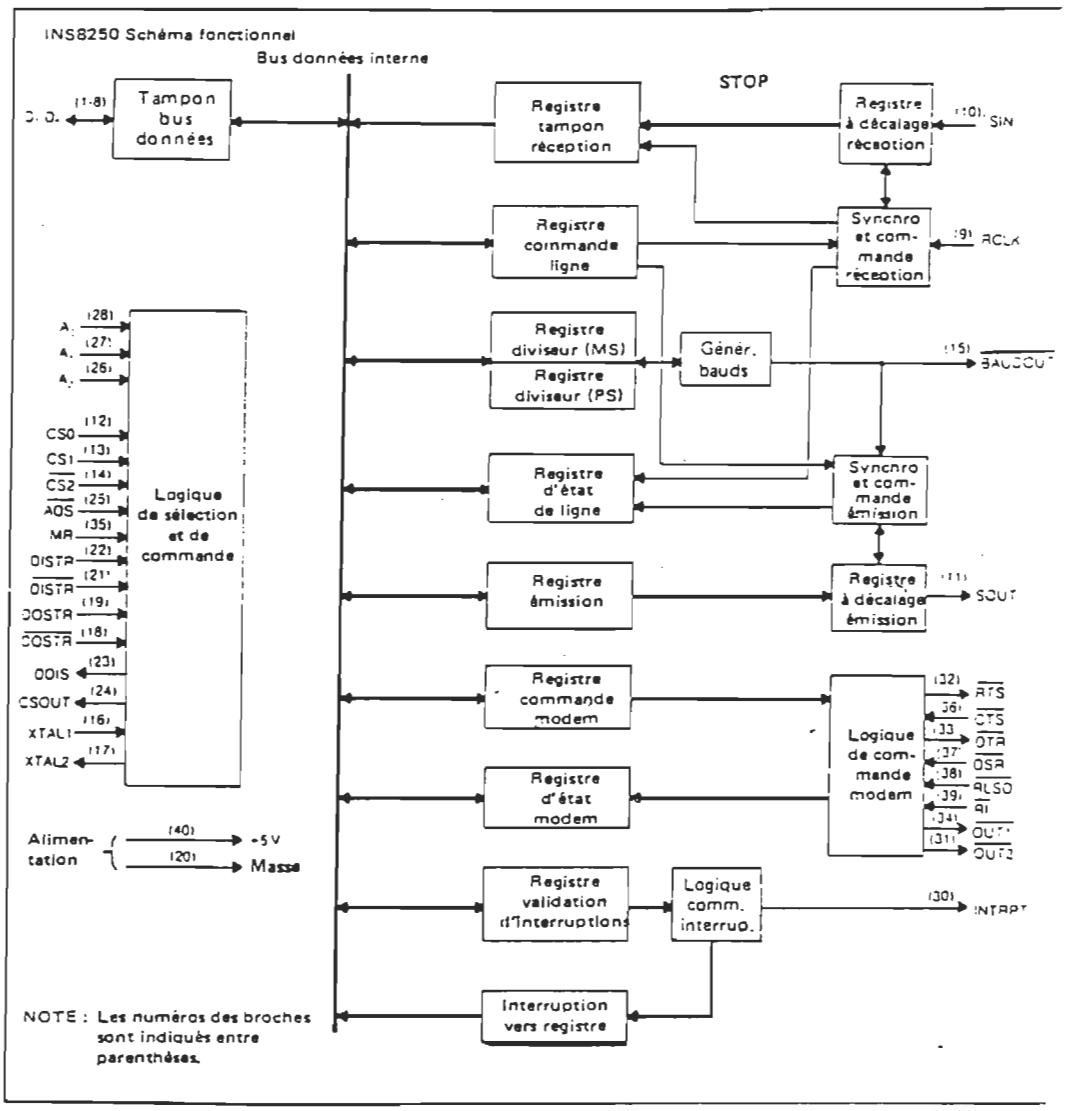


Schéma fonctionnel d'un ACE-8250

ANNEXE 6

MOTOROLA

MC3486

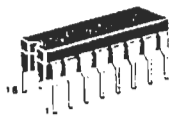
QUAD RS-422/423 LINE RECEIVER

Motorola's Quad RS-422/3 Receiver features four independent receiver chains which comply with EIA Standards for the Electrical Characteristics of Balanced/Unbalanced Voltage Digital Interface Circuits. Receiver outputs are 74LS compatible, three-state structures which are forced to a high impedance state when the appropriate output control pin reachs a logic zero condition. A PNP device buffers each output control pin to assure minimum loading for either logic one or logic zero inputs. In addition, each receiver chain has internal hysteresis circuitry to improve noise margin and discourage output instability for slowly changing input waveforms. A summary of MC3486 features include:

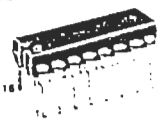
- Four Independent Receiver Chains
- Three-State Outputs
- High Impedance Output Control Inputs (PIA Compatible)
- Internal Hysteresis – 30 mV (Typ) @ Zero Volts Common Mode
- Fast Propagation Times – 25 ns (Typ)
- TTL Compatible
- Single 5.0 V Supply Voltage
- DS 3486 Provides Second Source

QUAD RS-422/3 LINE RECEIVER WITH THREE-STATE OUTPUTS

SILICON MONOLITHIC INTEGRATED CIRCUIT



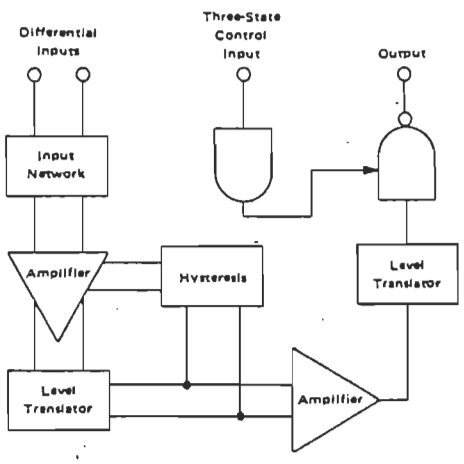
L SUFFIX
CERAMIC PACKAGE
CASE 620-02



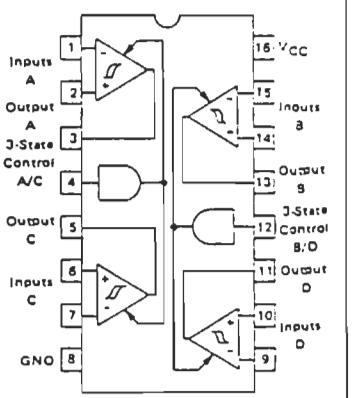
P SUFFIX
PLASTIC PACKAGE
CASE 648-05

5

RECEIVER CHAIN BLOCK DIAGRAM



PIN CONNECTIONS



ORDERING INFORMATION

DEVICE	TEMPERATURE RANGE	PACKAGE
MC3486L	0 to +70°C	Ceramic DIP
MC3486P	0 to +70°C	Plastic DIP

MOTOROLA

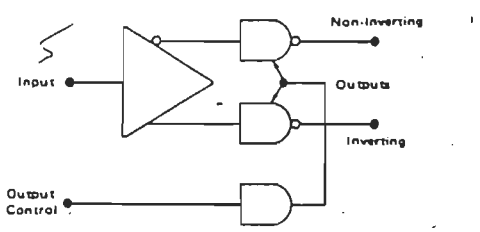
MC3487

QUAD LINE DRIVER WITH THREE-STATE OUTPUTS

Motorola's Quad RS-422 Driver features four independent driver chains which comply with EIA Standards for the Electrical Characteristics of Balanced Voltage Digital Interface Circuits. The outputs are three-state structures which are forced to a high impedance state when the appropriate output control pin reaches a logic zero condition. All input pins are PNP buffered to minimize input loading for either logic one or logic zero inputs. In addition, internal circuitry assures a high impedance output state during the transition between power up and power down. A summary of MC3487 features include:

- Four Independent Driver Chains
- Three-State Outputs
- PNP High Impedance Inputs (PIA Compatible)
- Fast Propagation Times (Typ 15 ns)
- TTL Compatible
- Single 5 V Supply Voltage
- Output Rise and Fall Times Less Than 20 ns
- DS 3487 Provides Second Source

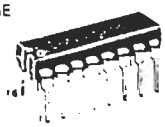
DRIVER BLOCK DIAGRAM



QUAD RS-422 LINE DRIVER WITH THREE-STATE OUTPUTS

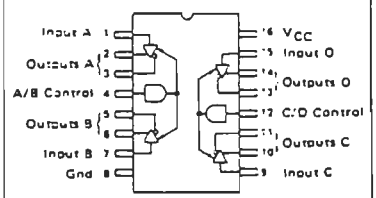
SILICON MONOLITHIC INTEGRATED CIRCUIT

L SUFFIX
CERAMIC PACKAGE
CASE 620-02



P SUFFIX
PLASTIC PACKAGE
CASE 648-05

PIN CONNECTIONS



TRUTH TABLE

Input	Control Input	Non-Inverting Output	Inverting Output
H	H	H	L
L	H	L	H
X	L	Z	Z

L = Low Logic State
H = High Logic State
X = Irrelevent
Z = Third-State (High Impedance)

MOTOROLA

MC1488

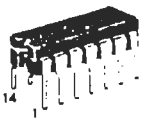
QUAD LINE DRIVER

The MC1488 is a monolithic quad line driver designed to interface data terminal equipment with data communications equipment in conformance with the specifications of EIA Standard No. RS-232C.

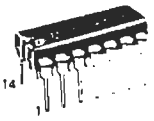
Features:

- Current Limited Output
  ±10 mA typ
- Power-Off Source Impedance
  300 Ohms min
- Simple Slew Rate Control with External Capacitor
- Flexible Operating Supply Range
- Compatible with All Motorola MDTL and MTTL Logic Families

QUAD MDTL LINE DRIVER
RS-232C
SILICON MONOLITHIC
INTEGRATED CIRCUIT

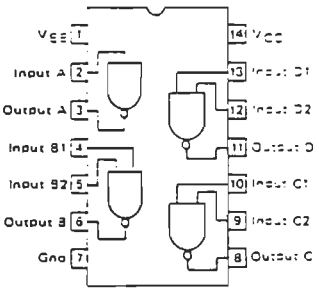


L SUFFIX
CERAMIC PACKAGE
CASE 632-02
MO-001AA

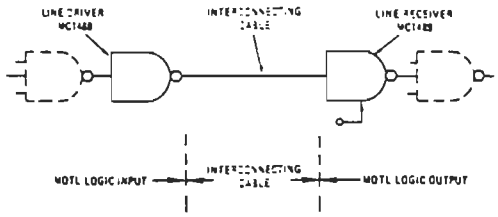


P SUFFIX
PLASTIC PACKAGE
CASE 646-03

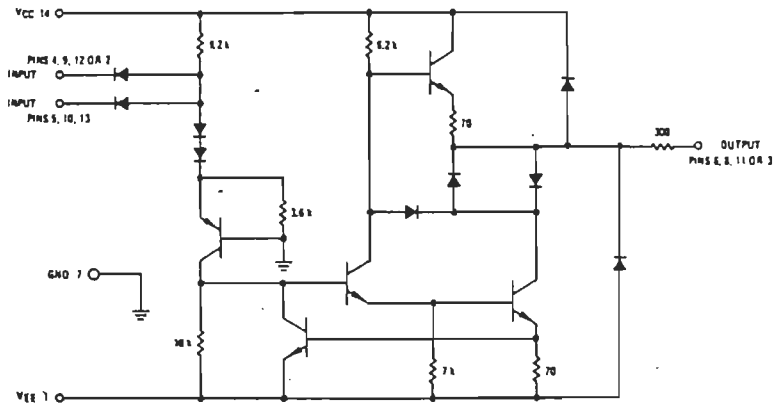
PIN CONNECTIONS



TYPICAL APPLICATION



CIRCUIT SCHEMATIC
(1/4 OF CIRCUIT SHOWN)



5

MOTOROLA

MC1489
MC1489A

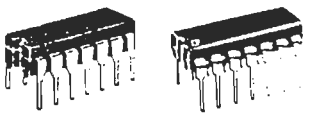
QUAD LINE RECEIVERS

The MC1489 monolithic quad line receivers are designed to interface data terminal equipment with data communications equipment in conformance with the specifications of EIA Standard No. RS-232C.

- Input Resistance – 3.0 k to 7.0 kilohms
- Input Signal Range – ±30 Volts
- Input Threshold Hysteresis Built In
- Response Control
  a) Logic Threshold Shifting
  b) Input Noise Filtering

QUAD MDTL
LINE RECEIVERS
RS-232C

SILICON MONOLITHIC
INTEGRATED CIRCUIT

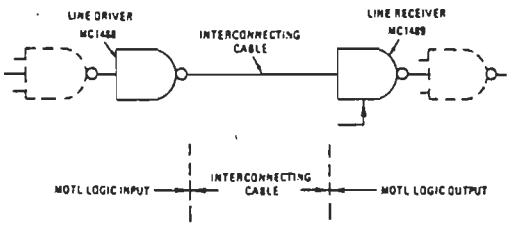


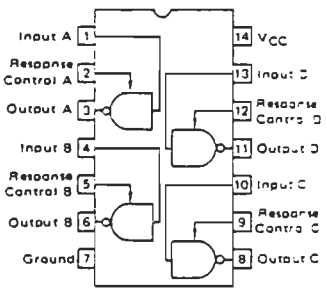
L SUFFIX
CERAMIC PACKAGE
CASE 632-02
MO-001AA

P SUFFIX
PLASTIC PACKAGE
CASE 646-06

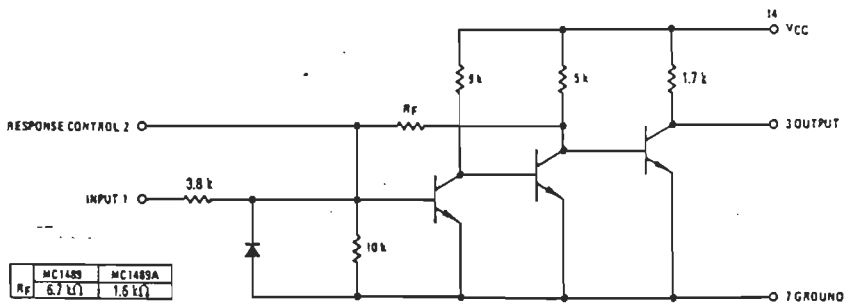
5

TYPICAL APPLICATION





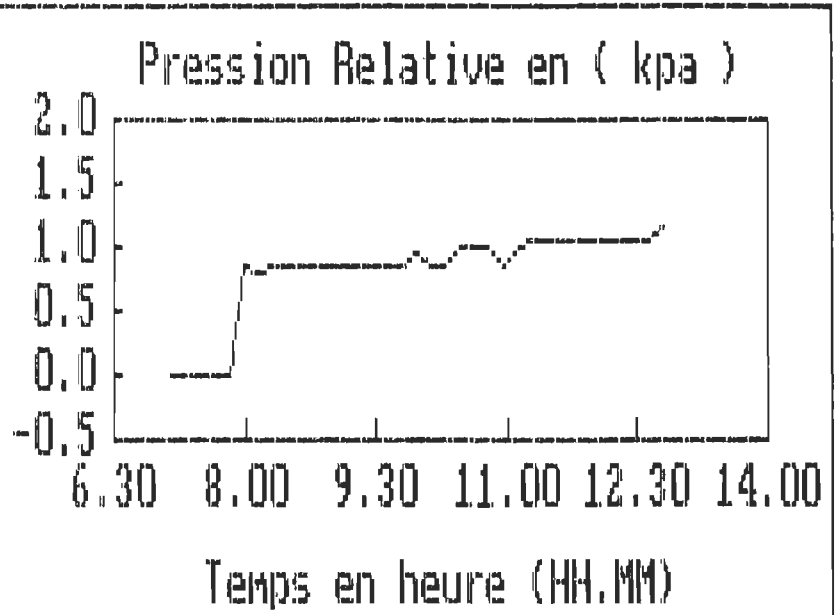
EQUIVALENT CIRCUIT SCHEMATIC (1/4 OF CIRCUIT SHOWN)

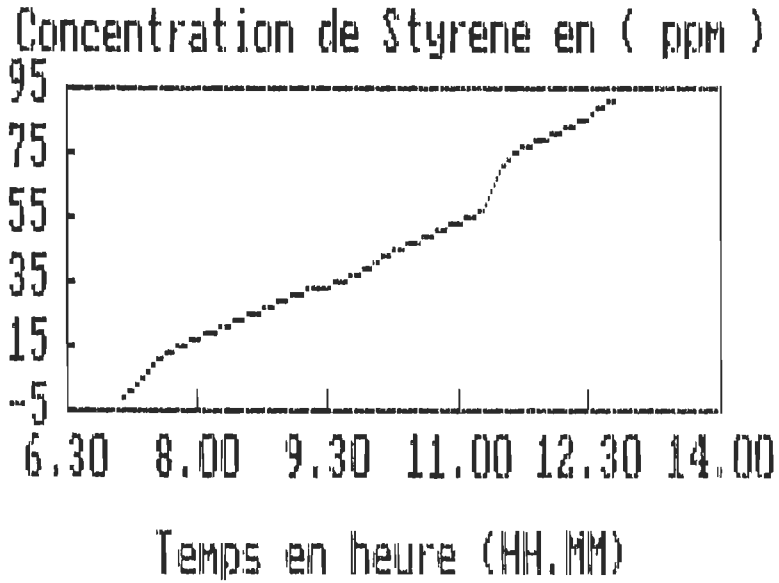


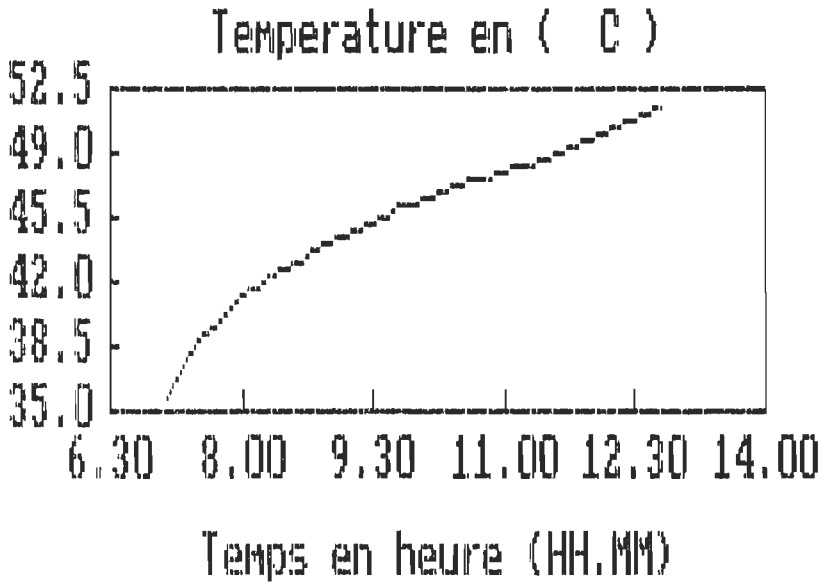
	MC1489	MC1489A
R_F	6.7 kΩ	1.6 kΩ

ANNEXE 7

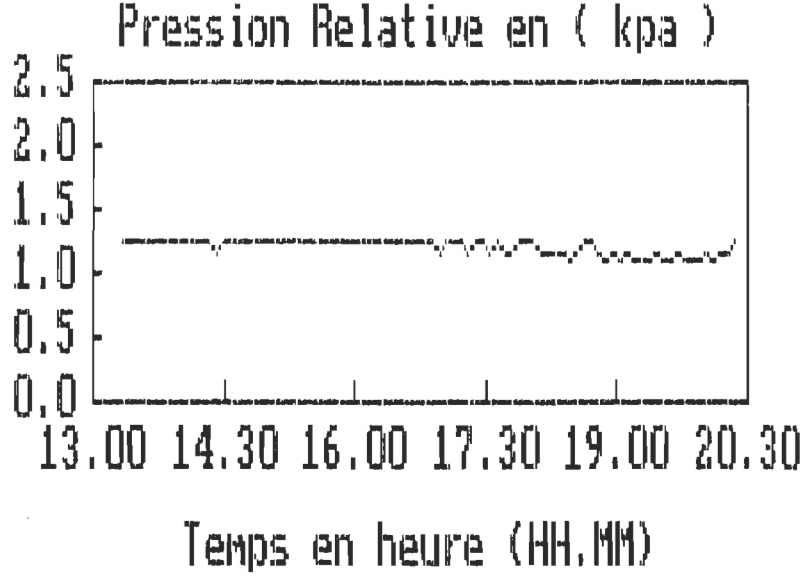
Experience : 1
Date : samedi Le 26.05.1991
La moyenne : 45.34 ppm

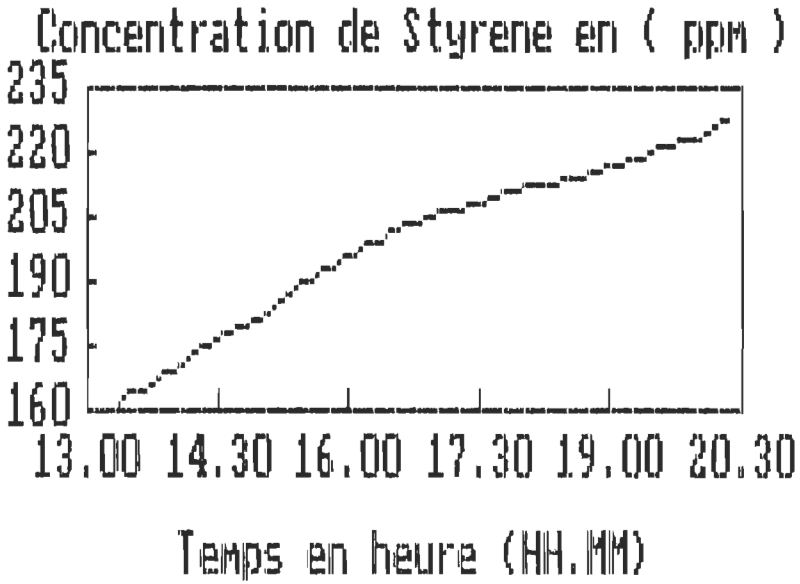


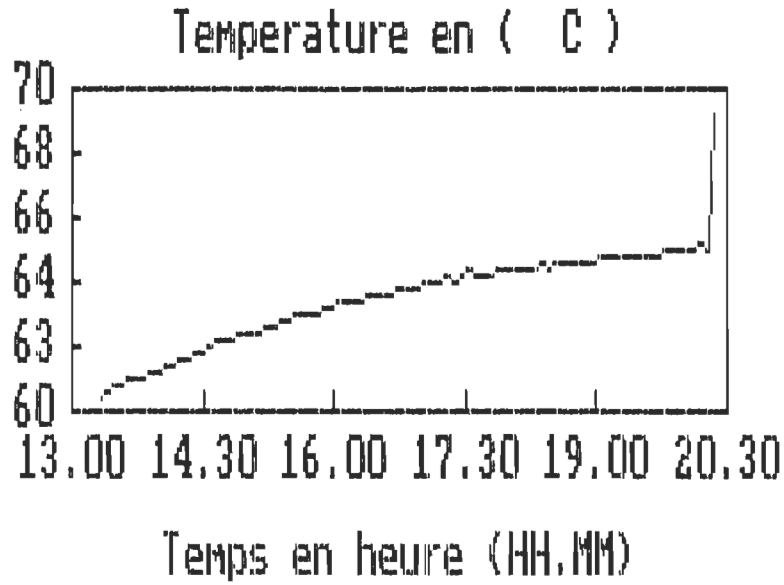




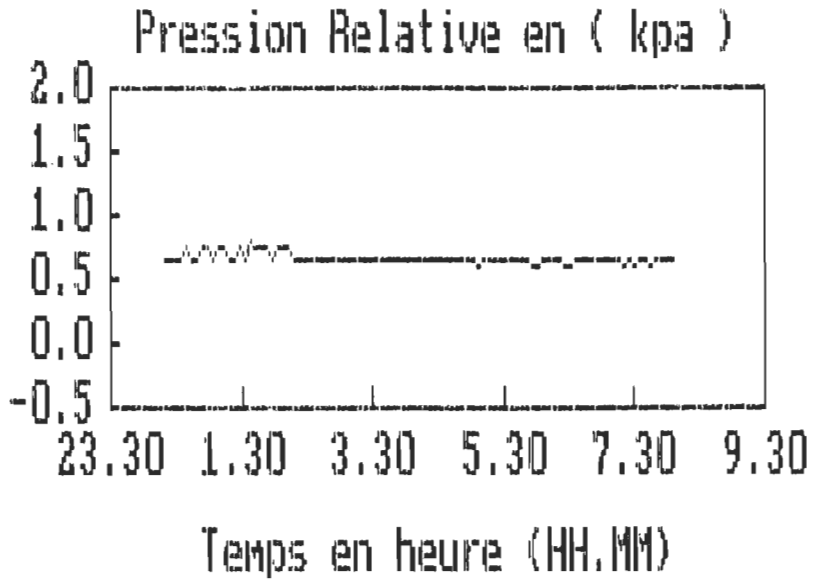
Experience : 2
Date : lundi Le 27.05.1991
La moyenne : 201.21 ppm

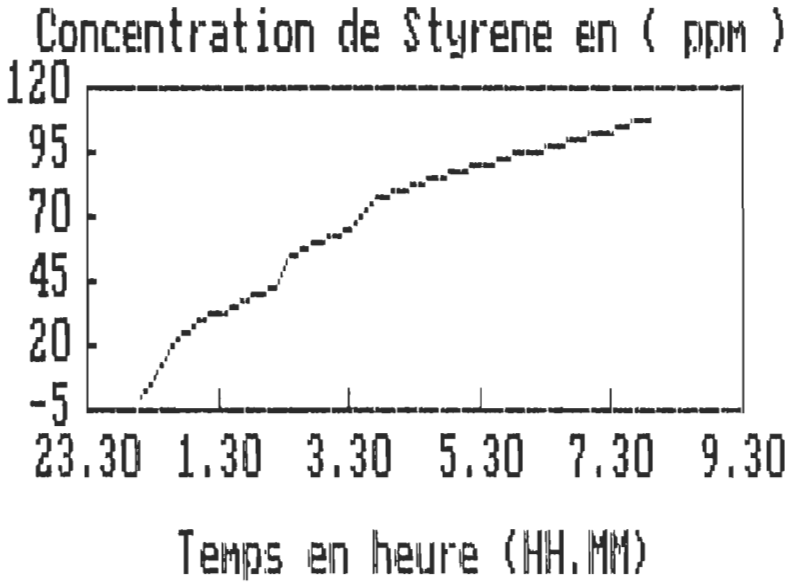


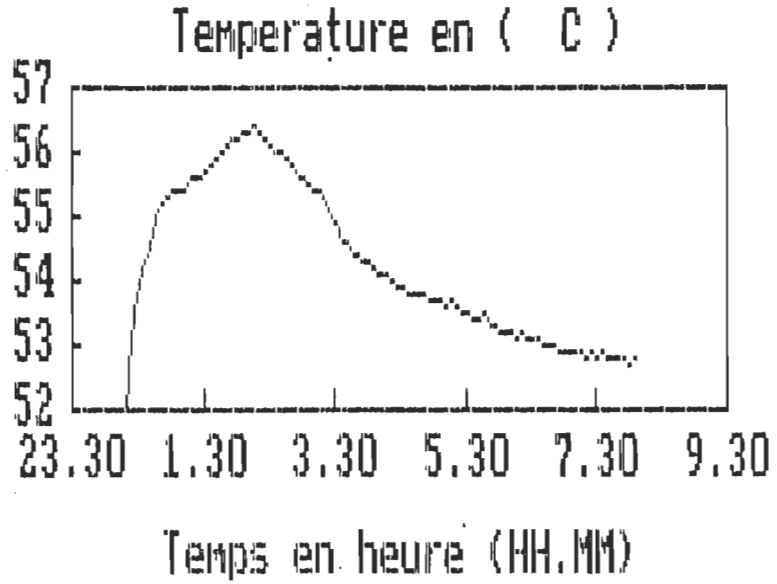




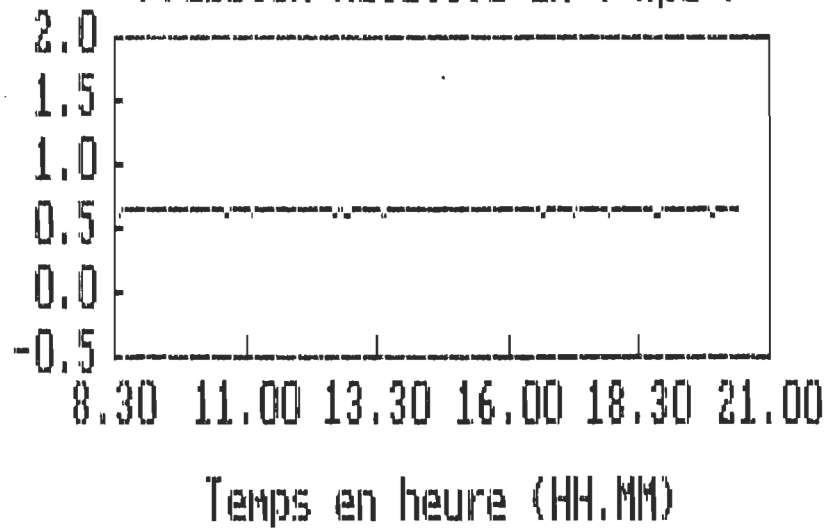
Experience : 3
Date : mardi Le 28.05.1991
La moyenne : 73.73 ppm

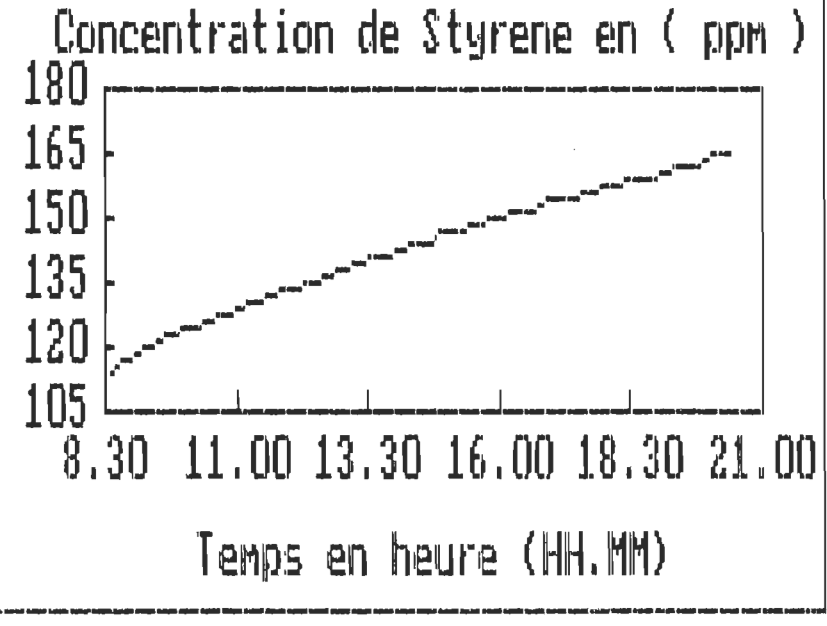


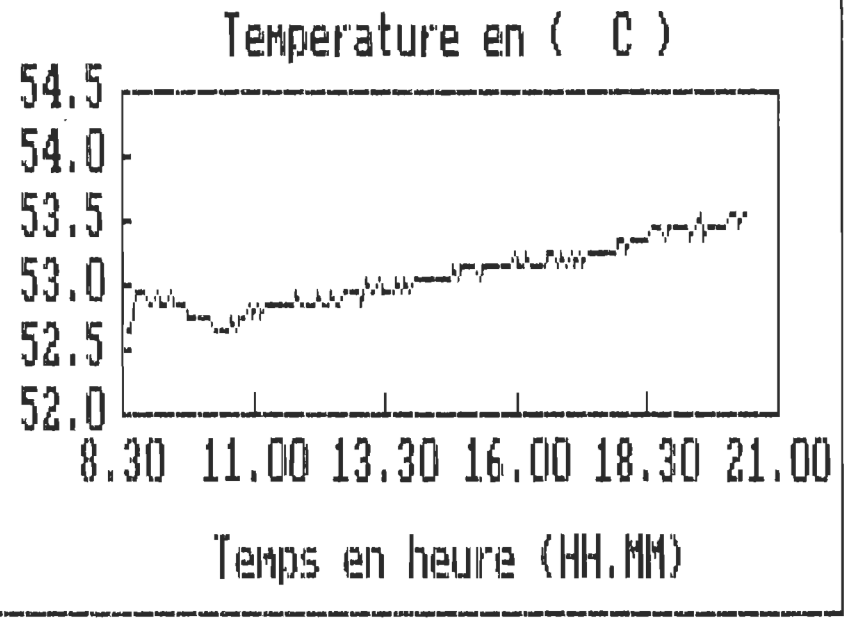




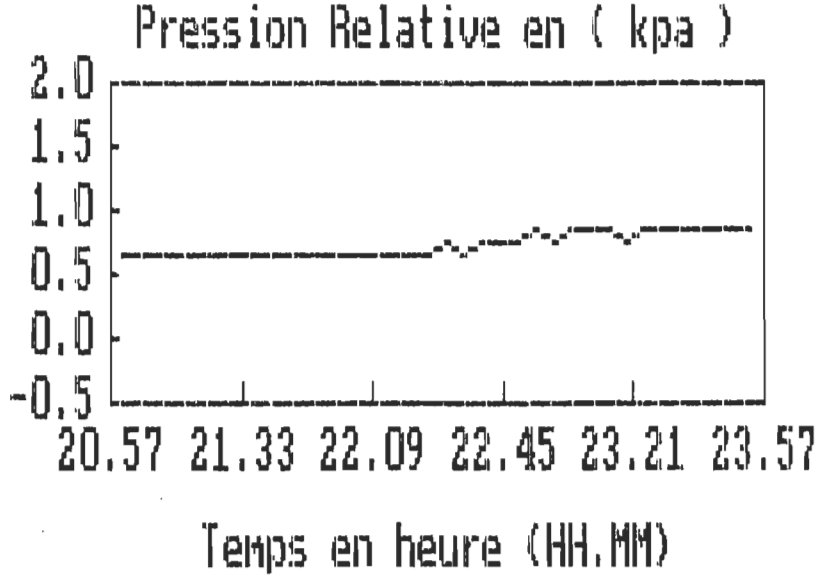
Experience : 4
Date : mercredi Le 29.05.1991
La moyenne : 144.35 ppm

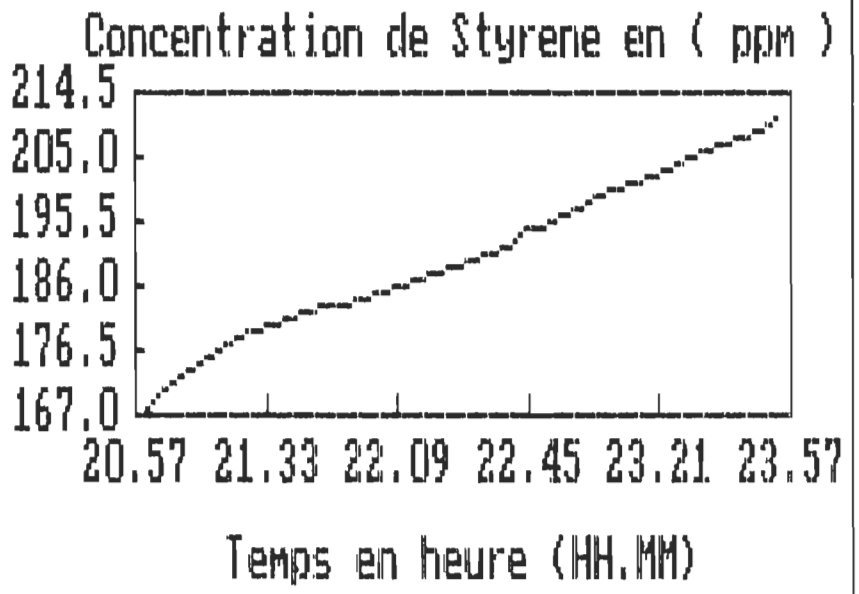


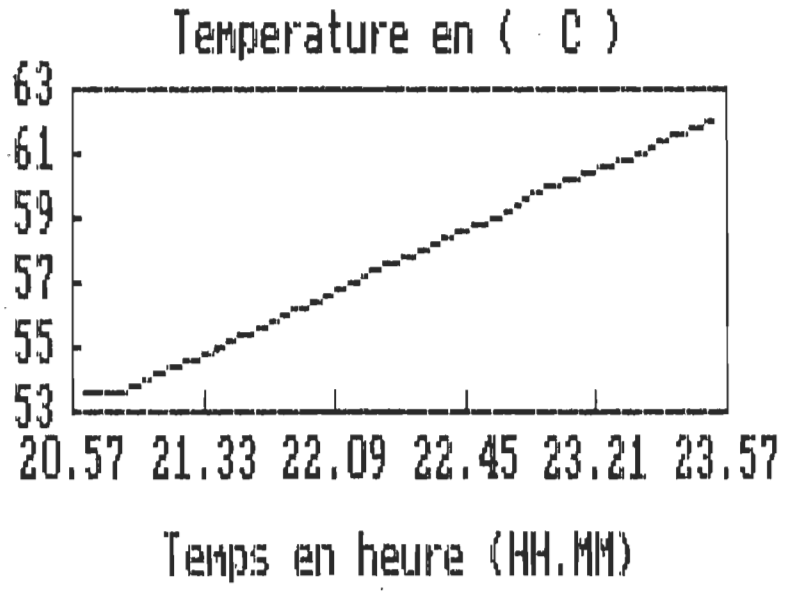




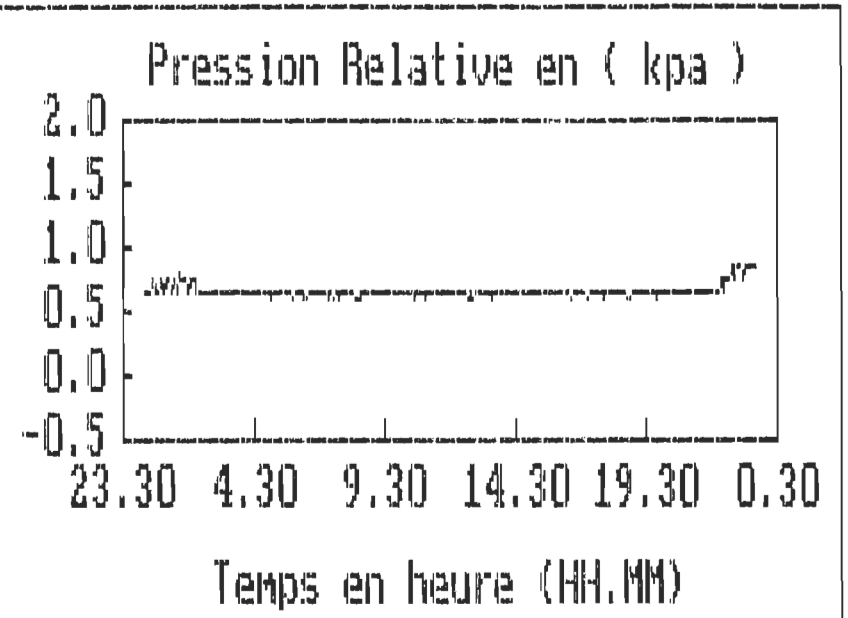
Experience : 5
Date : jeudi Le 30.05.1991
La moyenne : 191.64 ppm

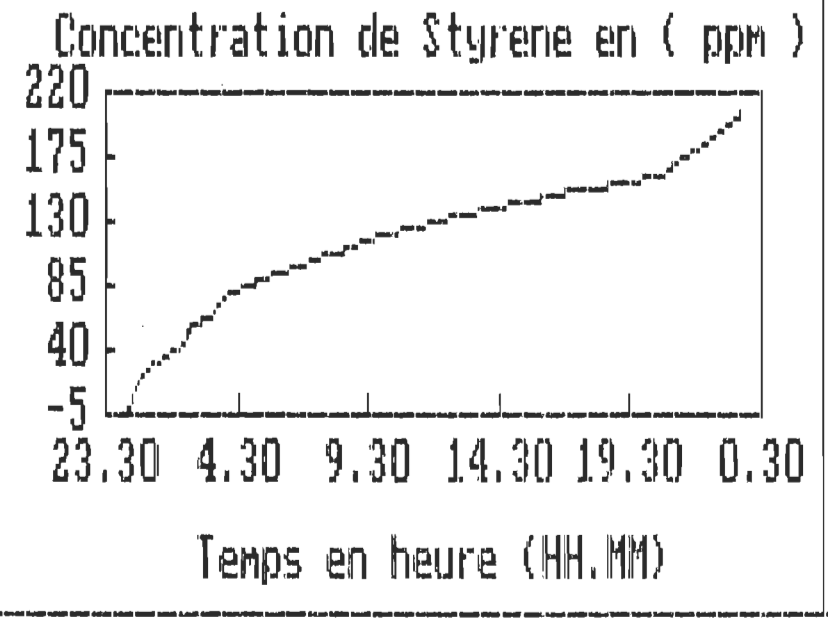


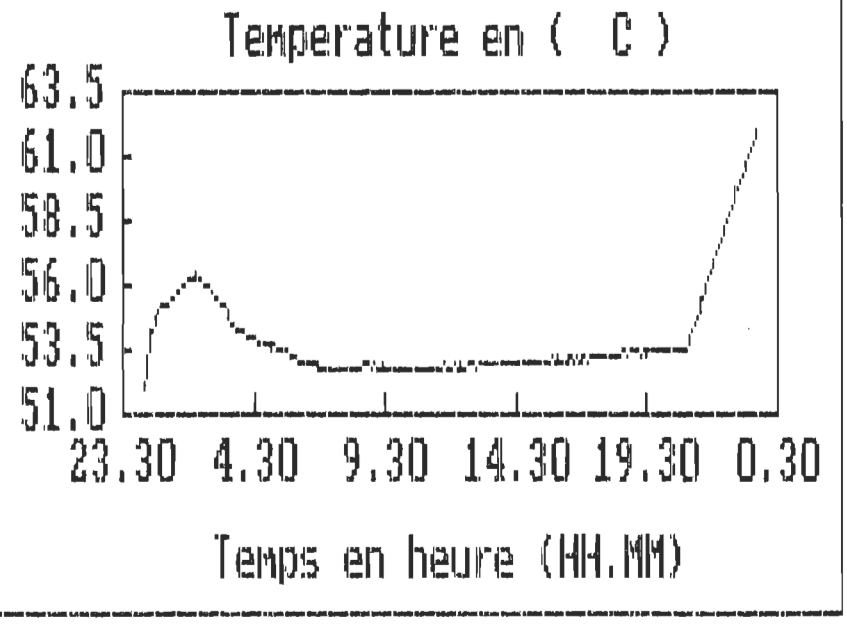




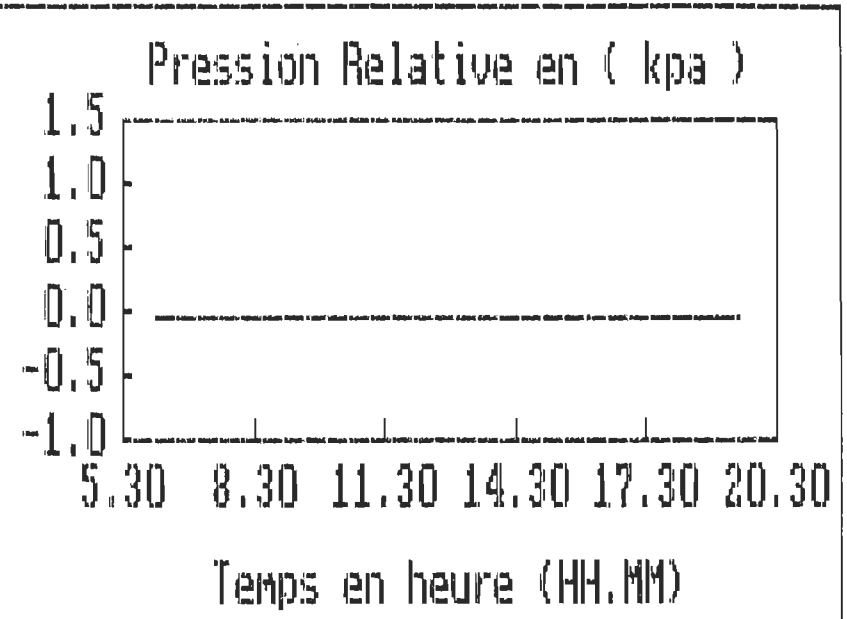
Experience : 6
Date : vendredi Le 31.05.1991
La moyenne : 126.00 ppm

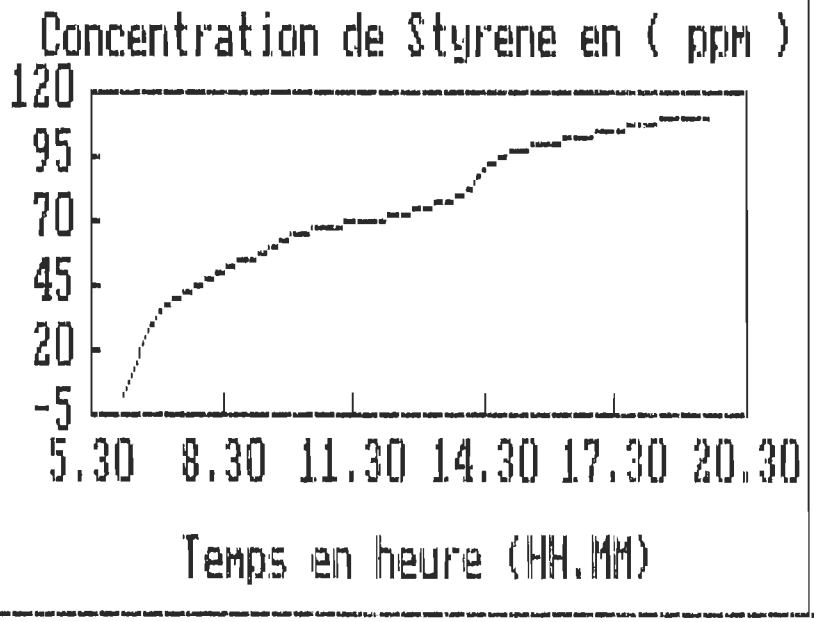


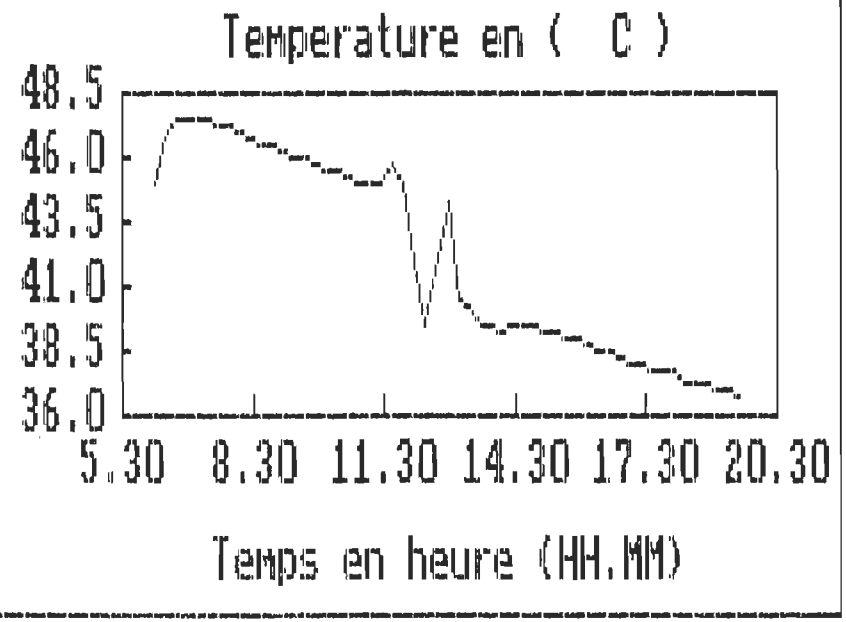




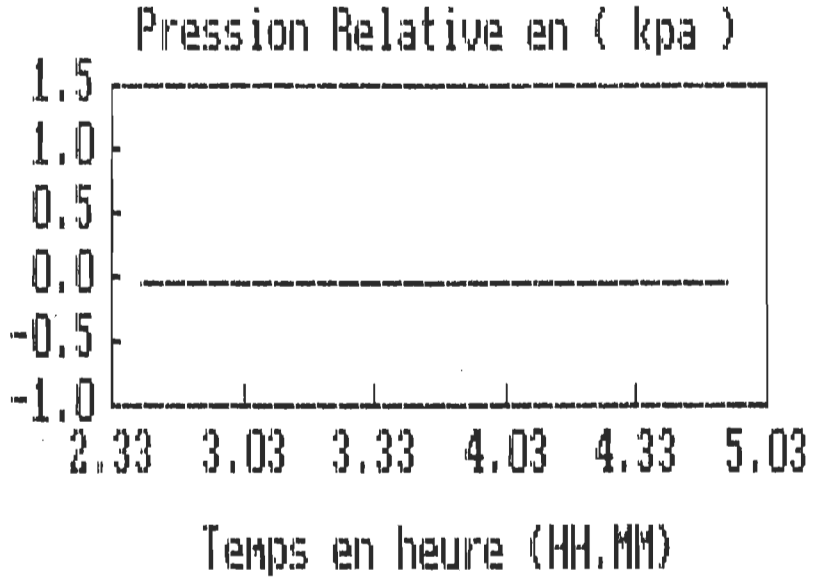
Experience : 7
Date : samedi Le 01.06.1991
La moyenne : 79.55 ppm

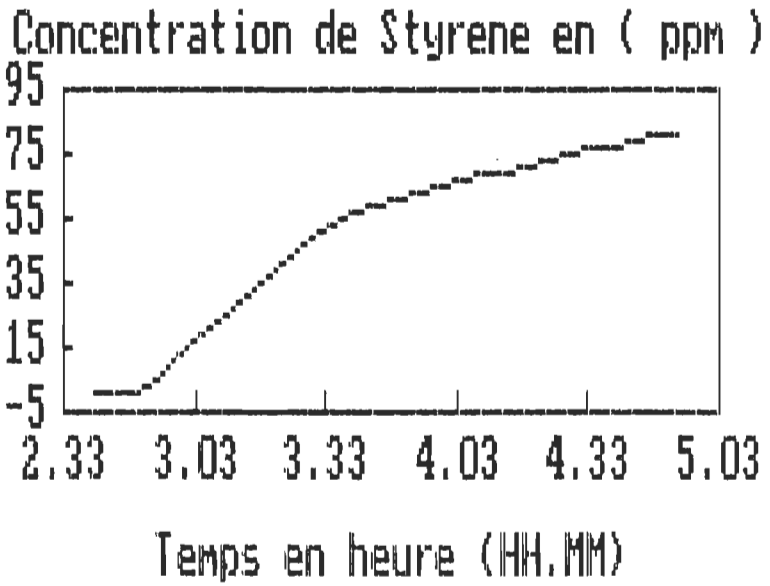


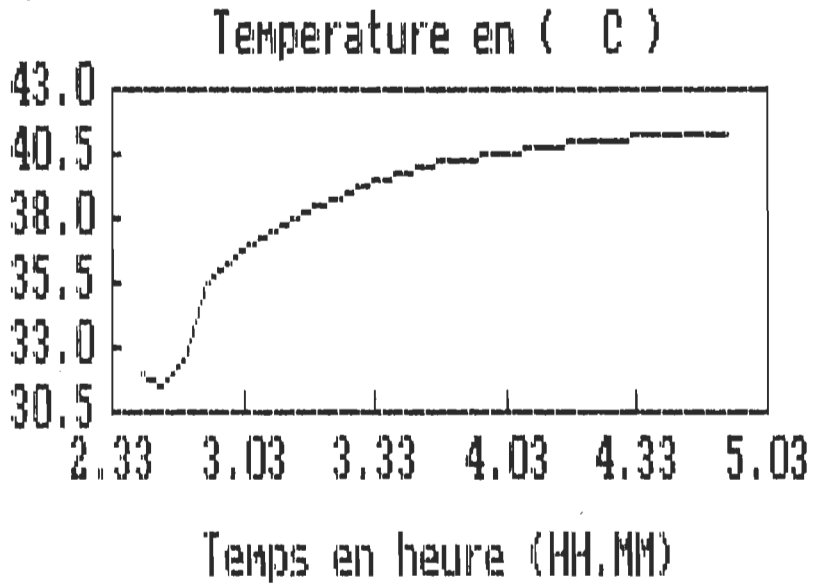




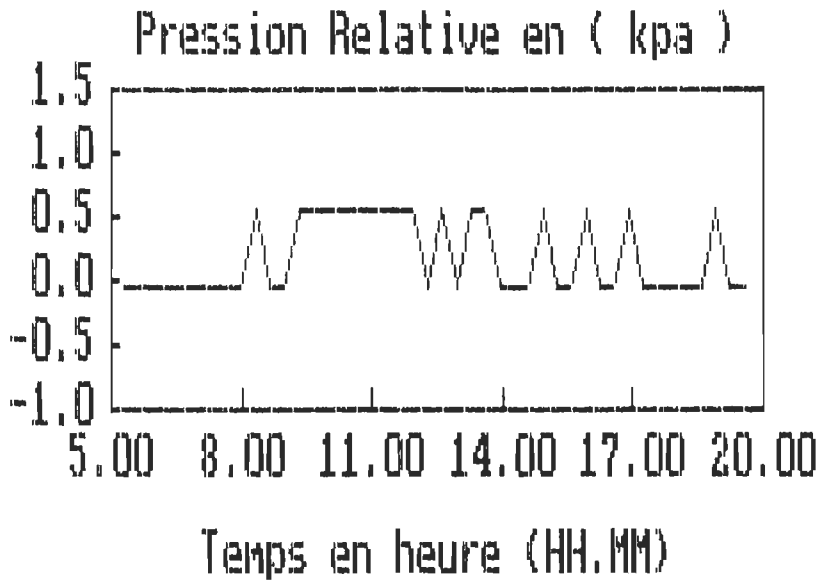
Experience : 8
Date : dimanche Le 02.06.1991
La moyenne : 47.03 ppm

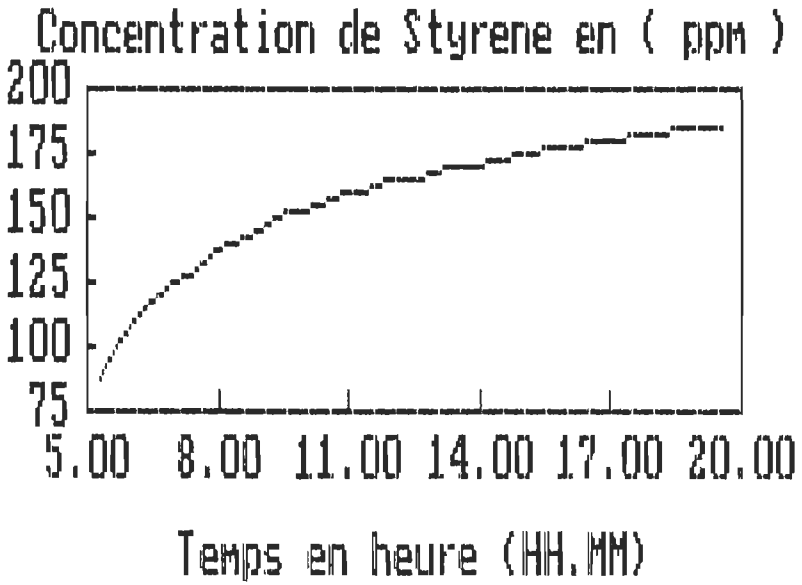


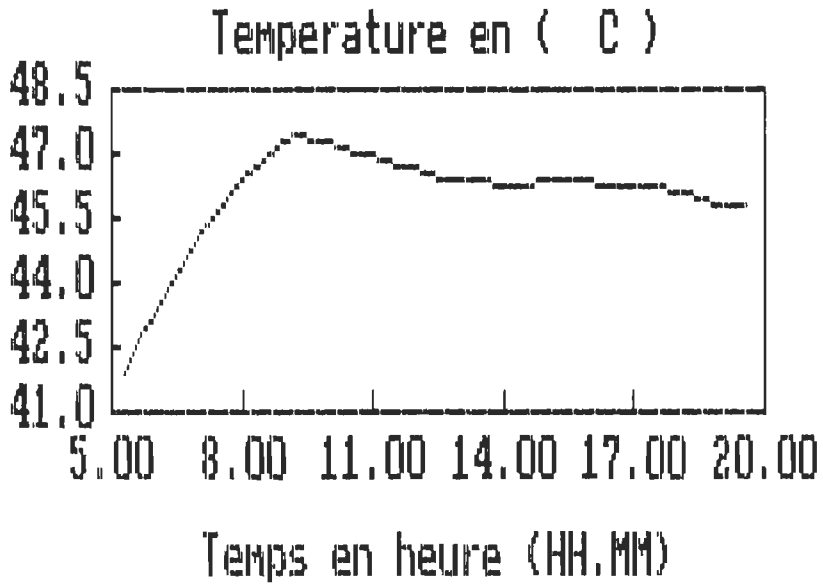




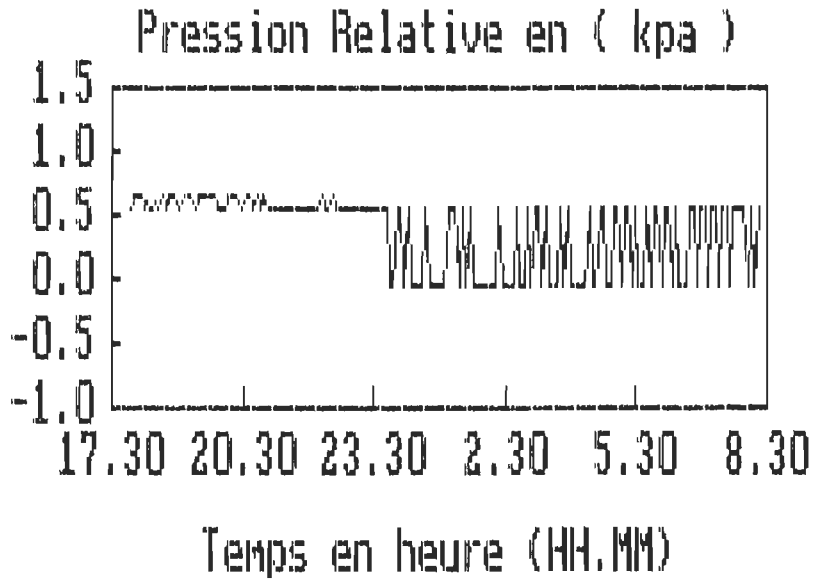
Experience : 9
Date : lundi Le 03.06.1991
La moyenne : 161.87 ppm

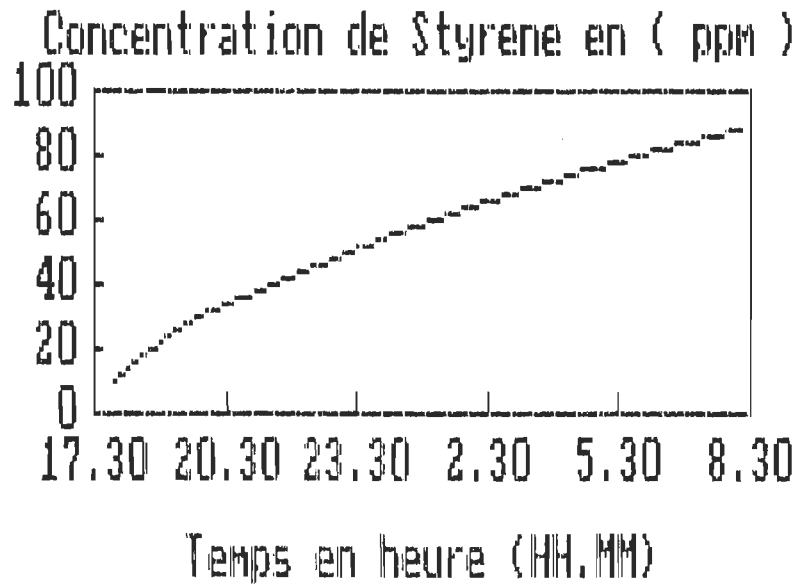


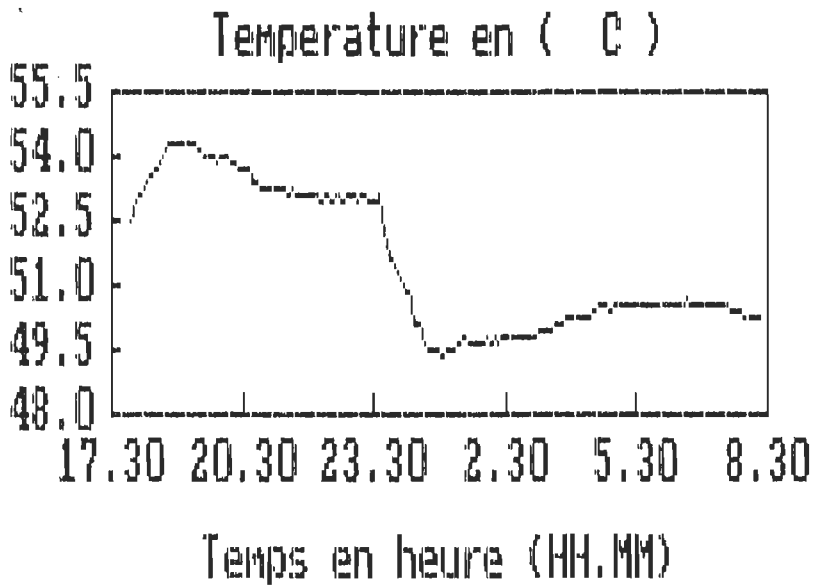




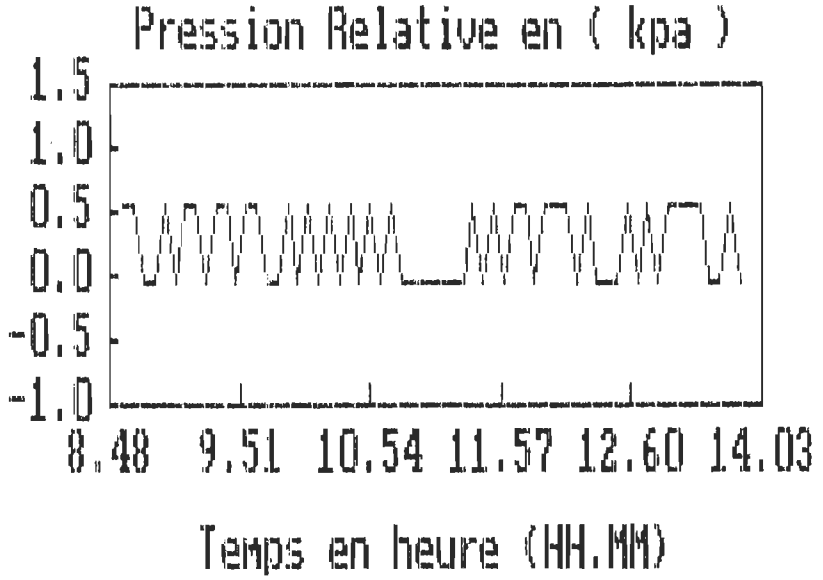
Experience : 10
Date : mardi Le 04.06.1991
La moyenne : 58.28 ppm

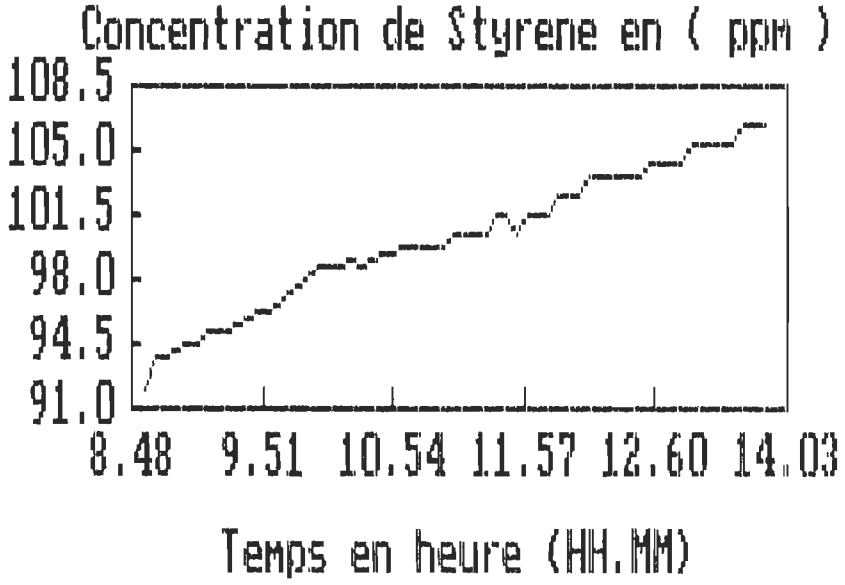


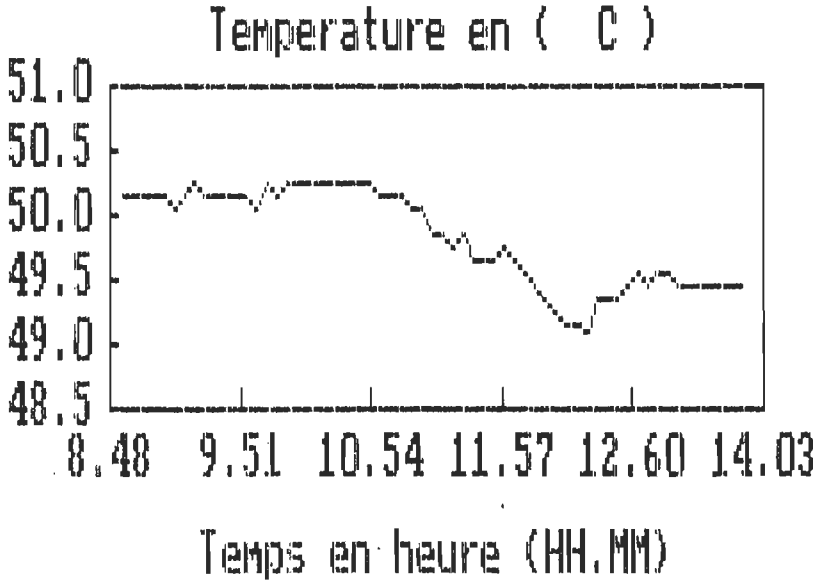




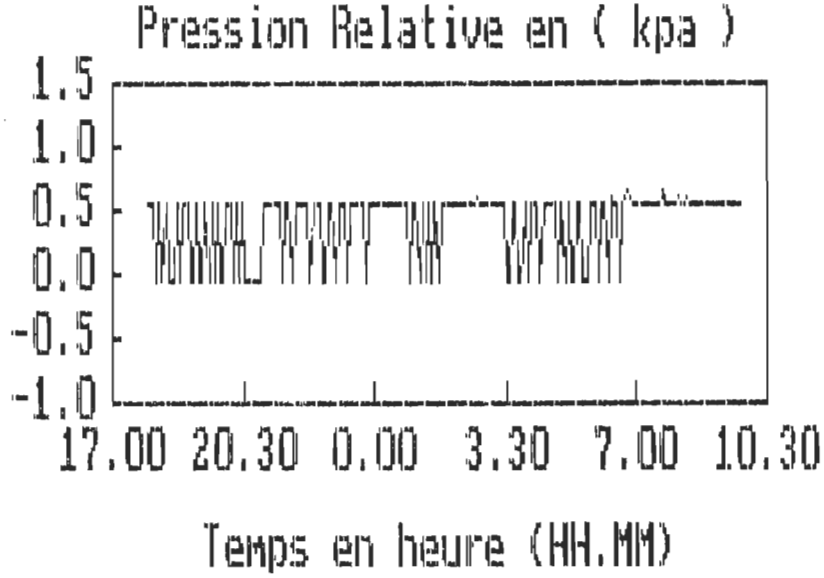
Experience : 11
Date : mercredi Le 05.06.1991
La moyenne : 100.02 ppm

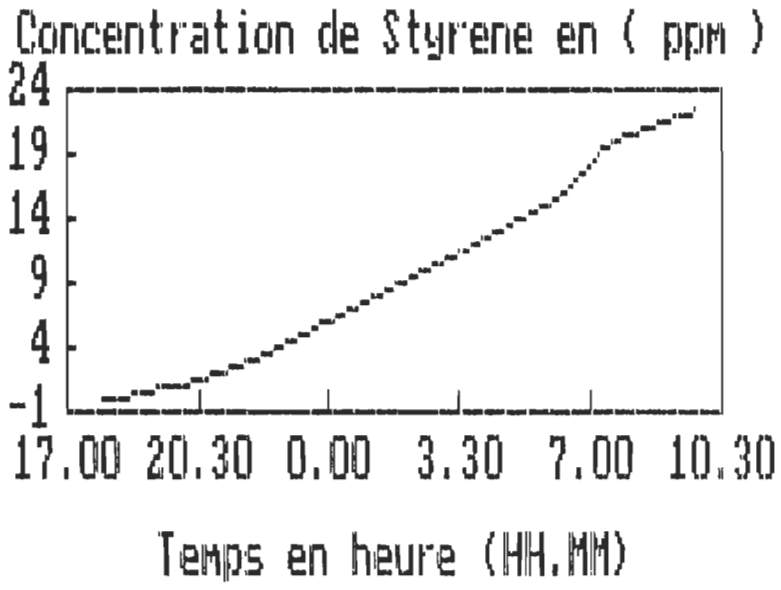


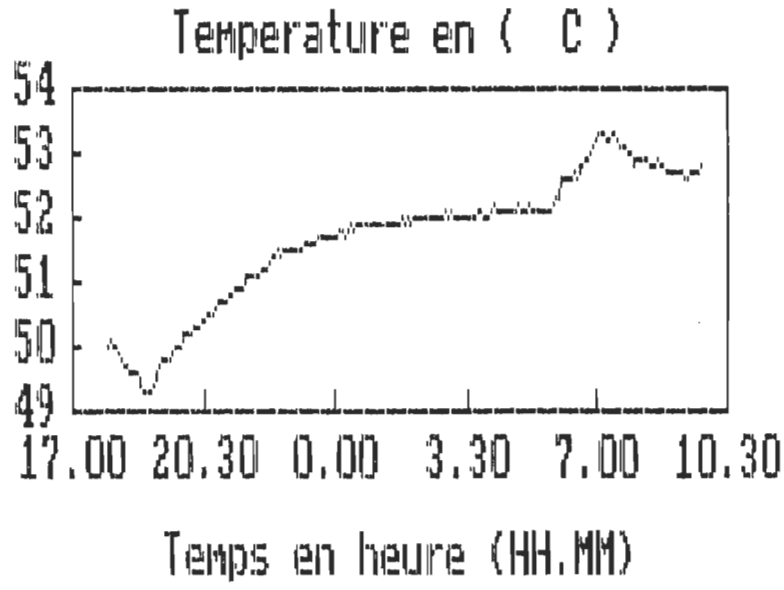




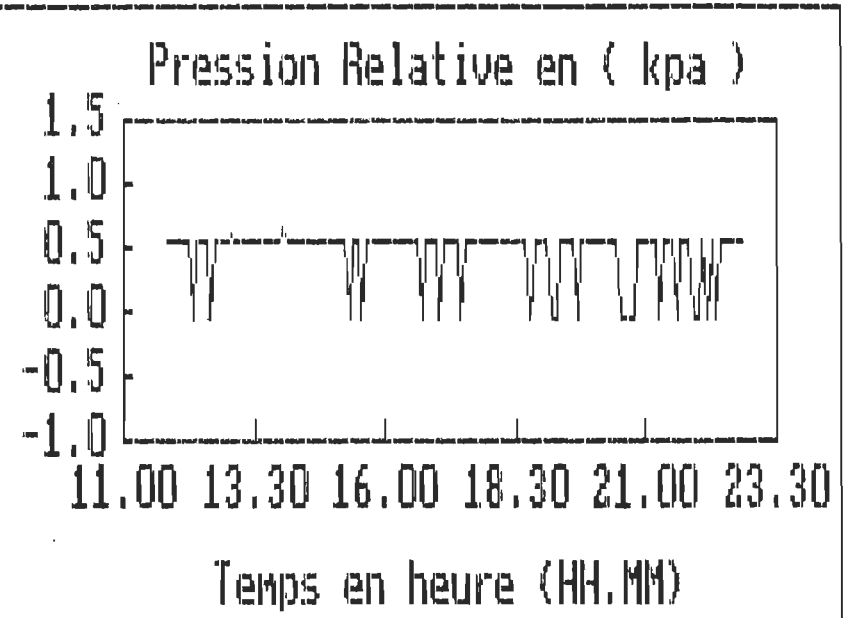
Experience : 12
Date : jeudi Le 06.06.1991
La moyenne : 9.77 ppm

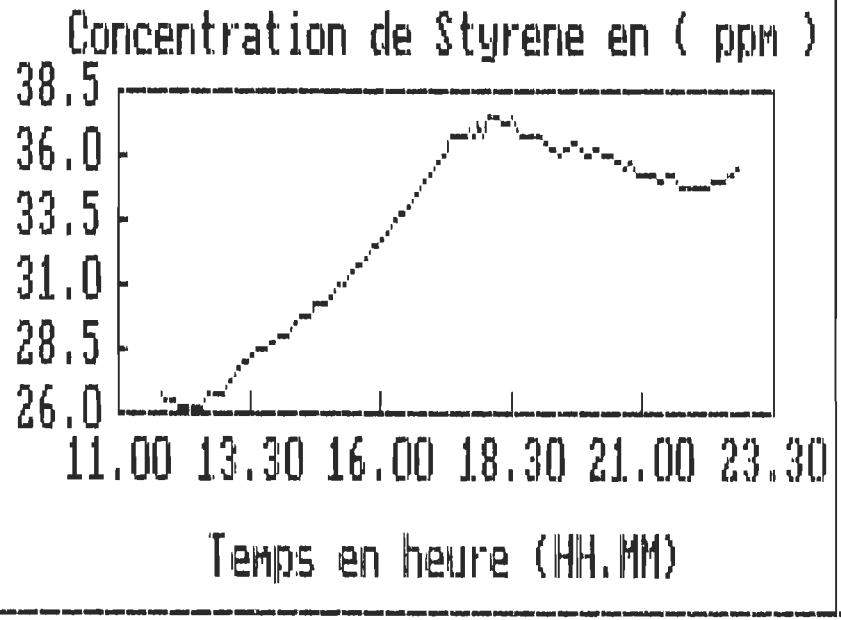


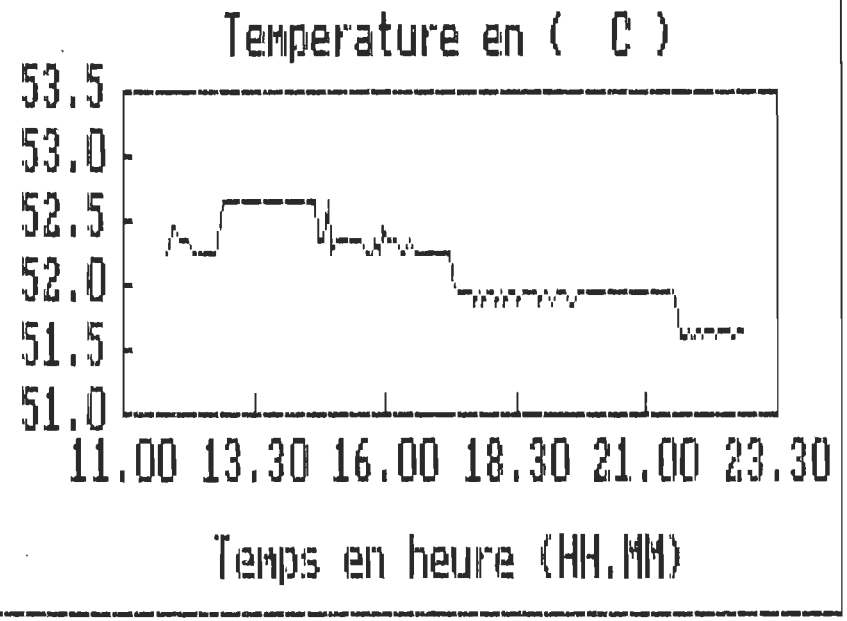




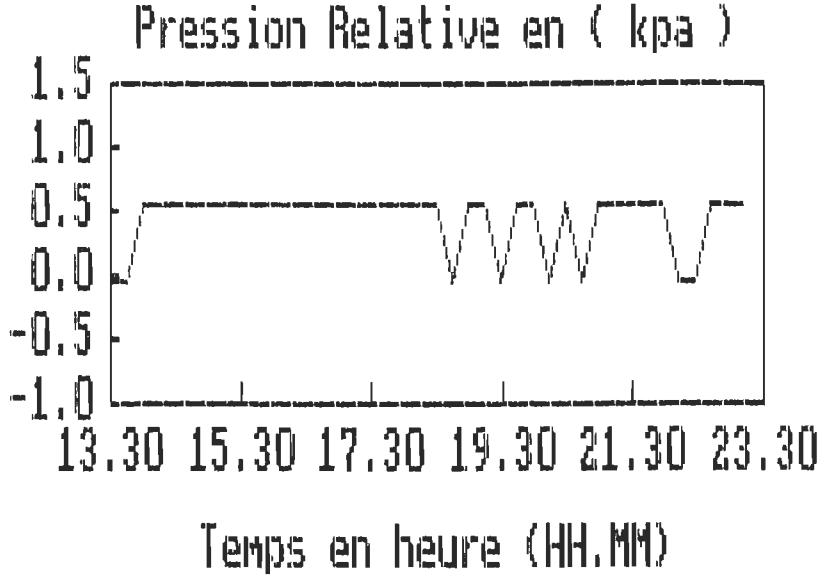
Experience : 13
Date : vendredi Le 07.06.1991
La moyenne : 33.21 ppm

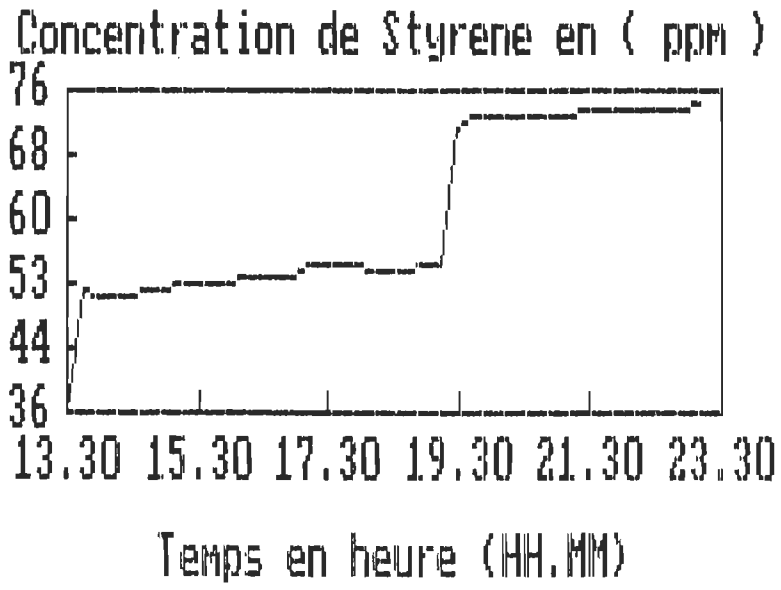


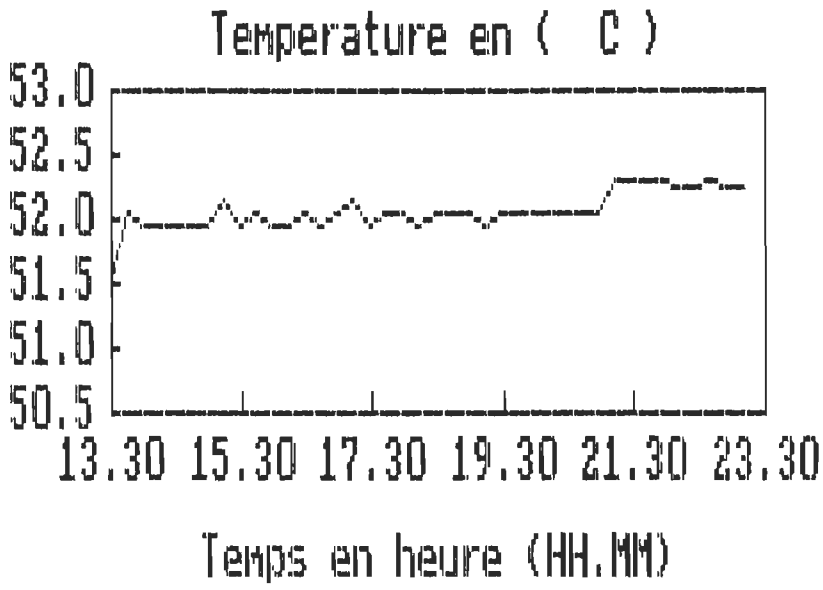




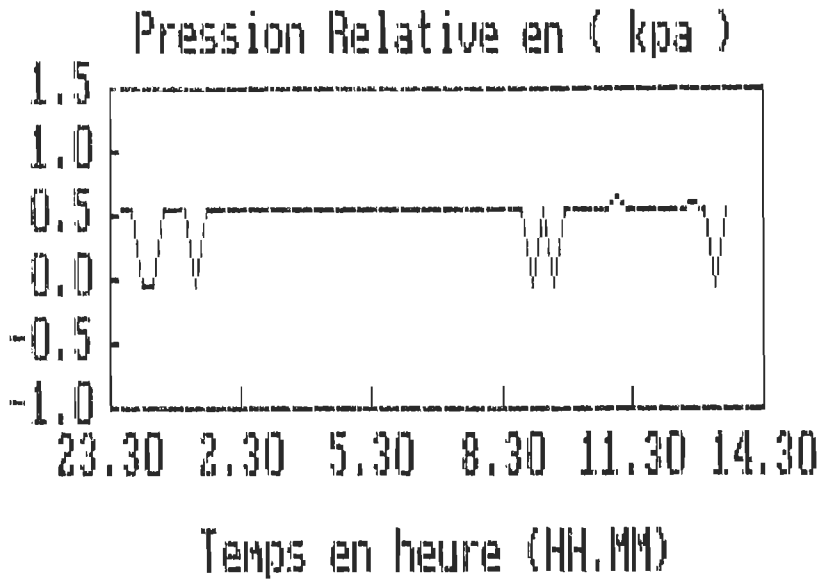
Experience : 14
Date : samedi Le 08.06.1991
La moyenne : 62.23 ppm

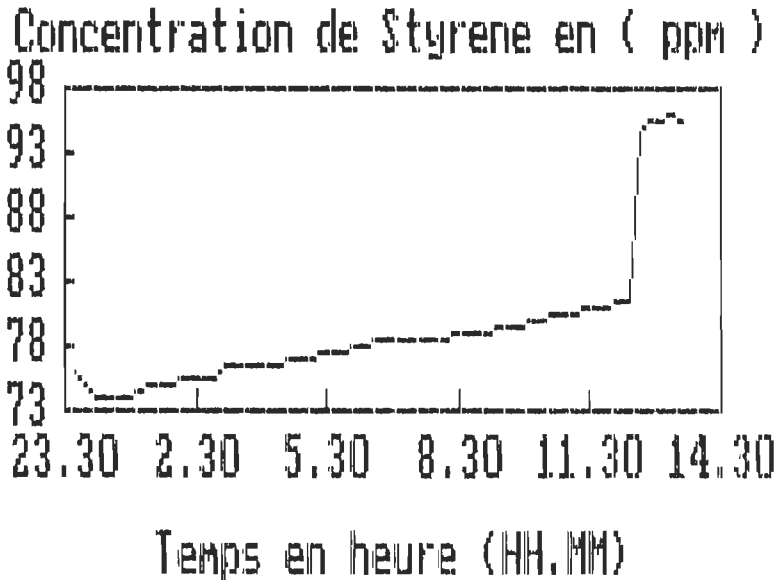


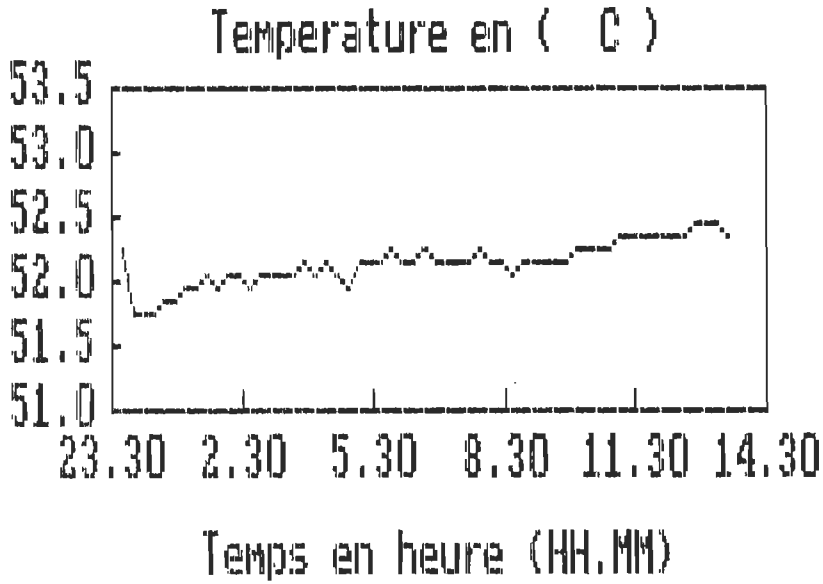




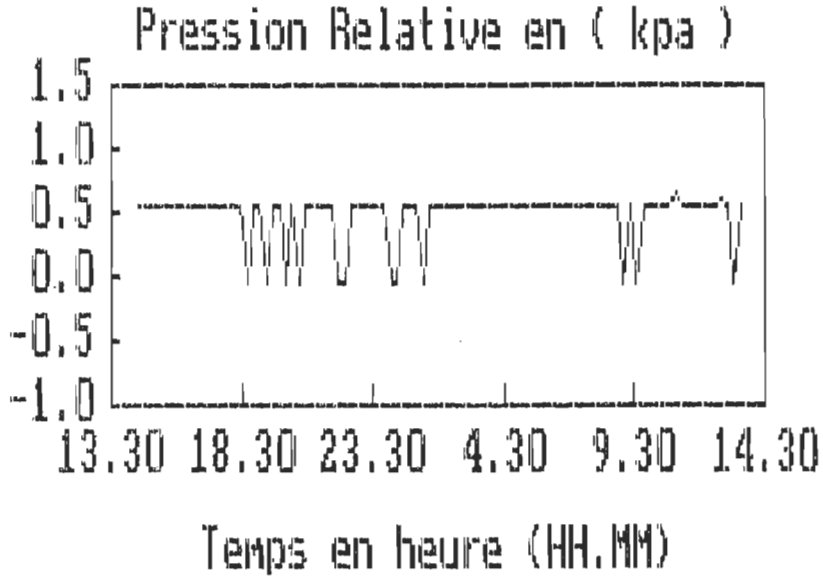
Experience : 15
Date : dimanche Le 09.06.1991
La moyenne : 79.21 ppm

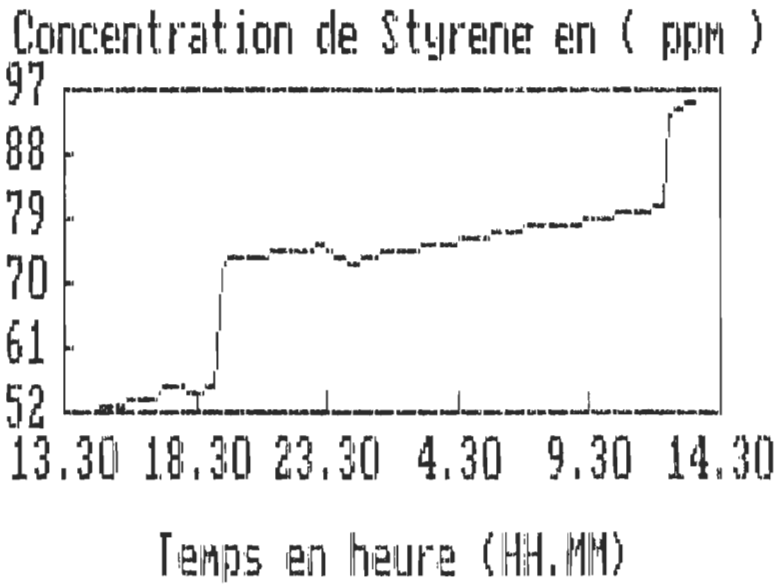


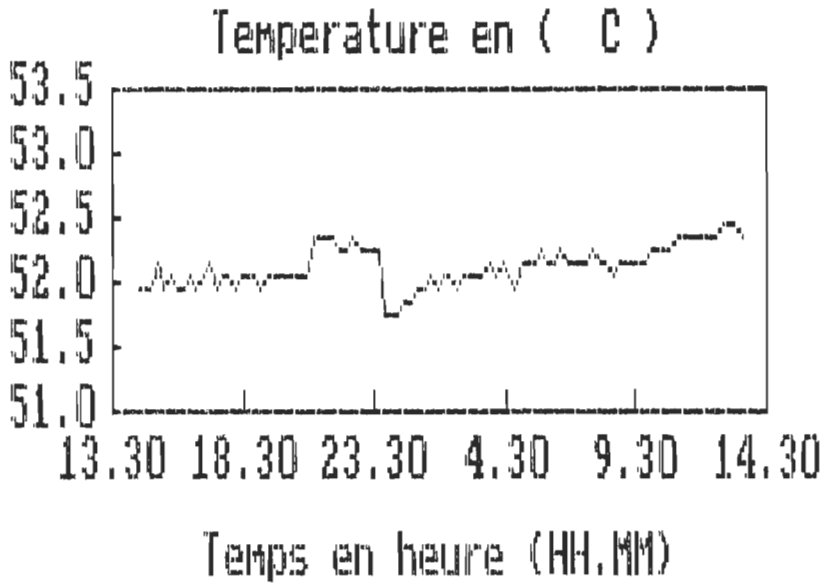




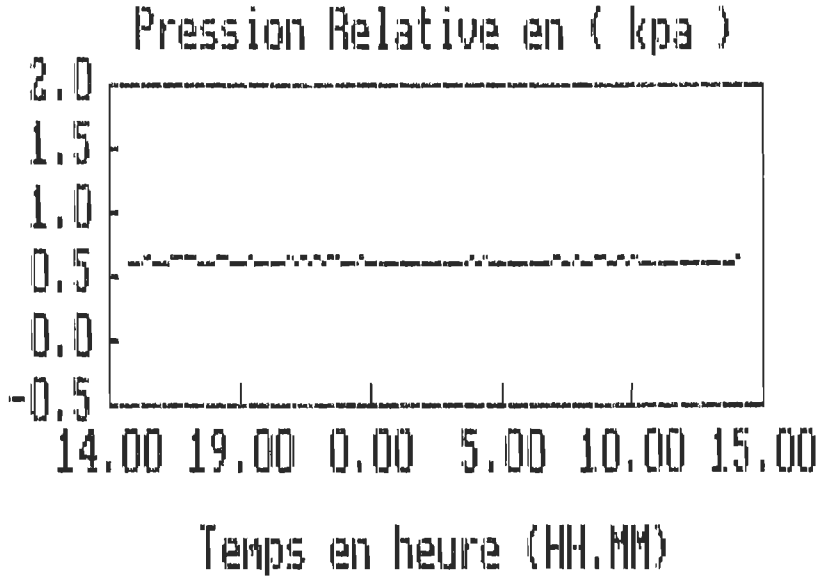
Experience : 16
Date : lundi Le 10.06.1991
La moyenne : 73.18 ppm

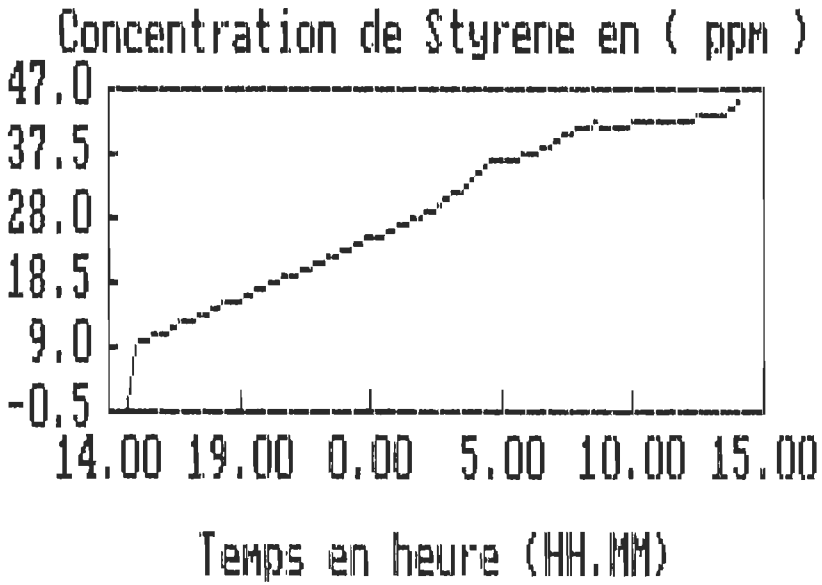


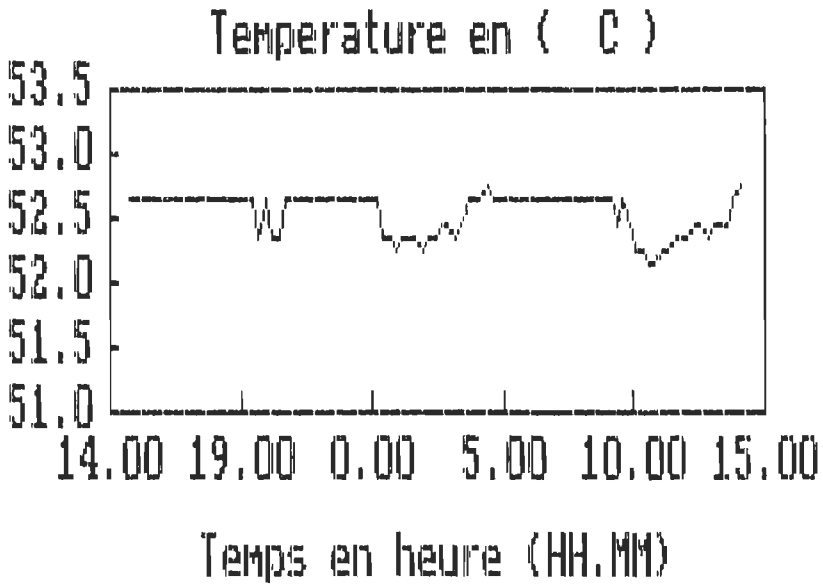




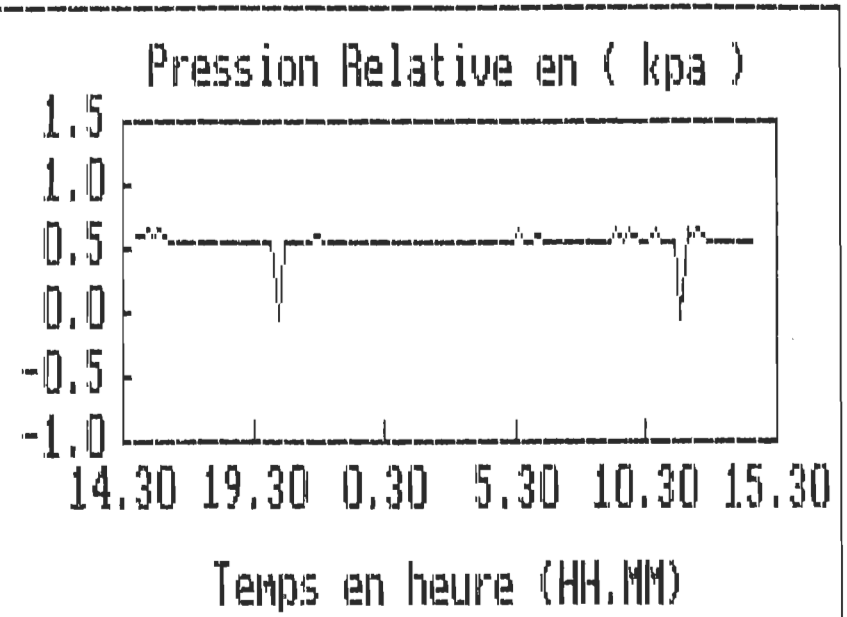
Experience : 17
Date : mardi Le 11.06.1991
La moyenne : 30.14 ppm

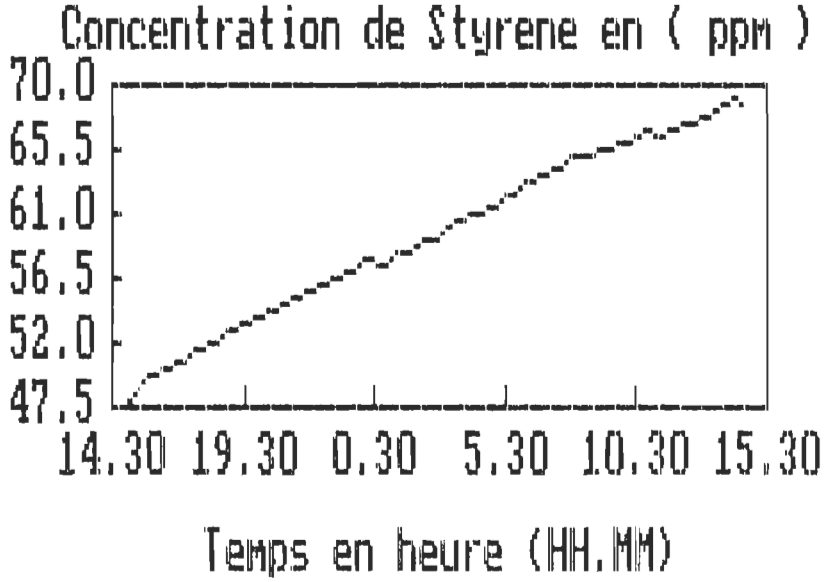


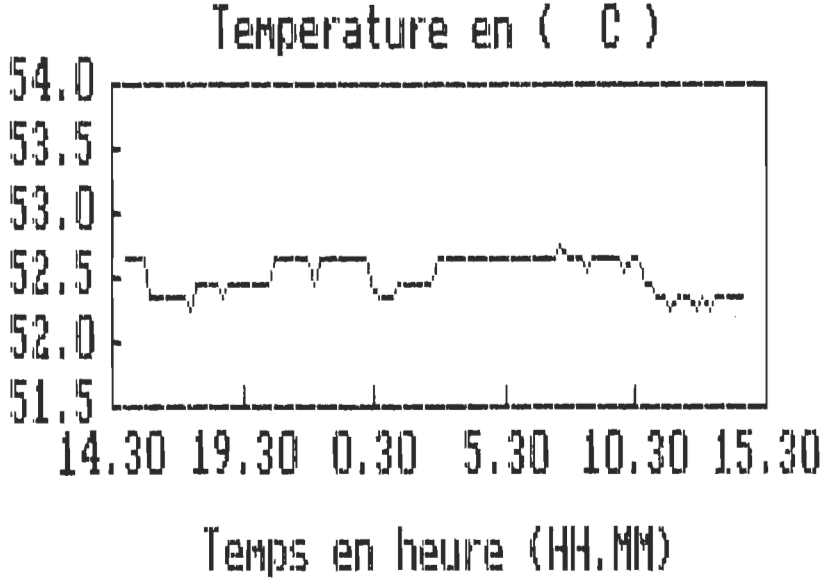




Experience : 18
Date : mercredi Le 12.06.1991
La moyenne : 59.96 ppm







Experience : 19
Date : jeudi Le 13.06.1991
La moyenne : 71.06 ppm